Desarrollo de una aplicación educativa para aprender Python

Fabian Bofill Diéguez

Grado de Ingeniería del software
Facultad de Informática

Universidad Complutense de Madrid

Curso 2020-2021
Director:
Antonio Sarasa Cabezuelo

Development of an educational application to learn Python

Fabian Bofill Diéguez

Grado de Ingeniería del software
Facultad de Informática

Universidad Complutense de Madrid

Curso 2020-2021
Director:
Antonio Sarasa Cabezuelo

A mis padres que más que apoyo son impulso.

AGRADECIMIENTOS

En primer lugar, dar las gracias a mi director Antonio Sarasa Cabezuelo, su guía y tutela durante el desarrollo de este trabajo tienen un valor incalculable. Los resultados obtenidos no serían posibles sin su ayuda.

Así mismo agradecer a mis padres Guillermo y Elvia todo el apoyo que me han brindado durante todo este proceso. Cuando mi motivación no era suficiente siempre he contado con la suya.

También agradecer a mi novia Paloma por hacer más fácil todo, acompañarme y por el apoyo moral que me ha dado.

RESUMEN

El proyecto consiste en una herramienta que permite generar cursos y caminos para el aprendizaje de un lenguaje de programación. Los caminos automatizan el contenido acorde a las evaluaciones obtenidas en los cursos.

La aplicación está contextualizada dentro del marco de las herramientas de aprendizaje móvil y traslada la mecánica de los juegos a un entorno educativo con el fin de conseguir mejores resultados. Mediante estas técnicas se busca aumentar la motivación y la participación durante la experiencia de aprendizaje.

Para la implementación, se ha desarrollado una aplicación web desde la cual se administrará el funcionamiento general. También se ha implementado una aplicación móvil que utiliza un servicio REST para obtener la información.

Palabras clave

Herramienta aprendizaje, cuestionarios, creación de cursos, aplicación web, aplicación móvil, API REST.

SUMMARY

The project is to build a tool that allows generating courses and itineraries for learning a programming language. The itineraries automate the content delivered according to the scores obtained in the courses.

The application falls within the framework of mobile learning tools and transfers the mechanics of games to an educational environment in order to achieve better results. These techniques seek to increase motivation and participation during the learning experience.

For implementation, a web application has been developed from which the general operation will be managed. A mobile application that uses a REST service to obtain the information has also been implemented.

Keywords

Learning tool, quizzes, course creation, web application, mobile application, REST API.

Índice de figuras y tablas

1. Introducción

En este capítulo de la memoria se explica la motivación de la aplicación desarrollada, sus objetivos y se describe la estructura de la memoria.

1.1. Motivación

Mantener la motivación y el interés durante el proceso de aprendizaje es muchas veces complicado, más aún cuando se recurre a metodologías convencionales. Estas se centran en la memorización de conceptos y siguen un itinerario único para todos los alumnos.

La utilización de elementos de juegos en el ámbito educativo es cada vez más frecuente. Esta metodología gana terreno como medio de formación ya que facilita el aprendizaje, aumenta la motivación y la participación.

Los lenguajes de programación no quedan exentos de poder utilizar este método que permite obtener retroalimentación de forma constante. Como para el aprendizaje no se puede utilizar un producto en el que encaje todas las personas siguiendo el mismo camino independientemente del nivel o de la velocidad del alumnado, sería conveniente poder generar caminos distintos que se desarrollen teniendo en cuenta los resultados que se obtienen durante la realización de los cursos.

En este trabajo se ha desarrollado una herramienta que implementa un sistema de gestión de cursos para aprender a programar en Python. Este sistema permite a los profesores crear y gestionar cursos que formarán los caminos que los alumnos seguirán. El sistema facilita que los alumnos sigan dichos caminos.

1.2. Objetivos

El objetivo de este proyecto es desarrollar una herramienta que permita generar los cursos y crear caminos a seguir, según las puntuaciones obtenidas. Todo ello haciendo uso de las ya mencionadas mecánicas de juegos. Los objetivos específicos planteados fueron los siguientes:

- Desarrollar una aplicación web que permita a los profesores crear los cursos, las preguntas y estructurar los itinerarios o caminos a seguir. Además debería poder obtener información del desempeño de los alumnos inscritos en los cursos.

- Implementar una API REST que contenga la lógica necesaria para recibir todas las peticiones de la aplicación móvil.

- Desarrollar una aplicación móvil Android mediante la cual los alumnos puedan utilizar la herramienta para completar los cursos.

1.3. Estructura de la memoria

La memoria de este proyecto se estructura de la siguiente manera. En el capítulo 1 se introduce el trabajo realizado y la motivación para el desarrollo. Además se describe la estructura de la memoria.

En el capítulo 2 se comentan herramientas con una funcionalidades semejante a la desarrollada en este proyecto.

El capítulo 3 detalla la tecnología utilizada para el desarrollo.

El capítulo 4 especifica los casos de uso de la aplicación divididos en 3 módulos: profesores, alumnos y comunes.

El capítulo 5 trata la forma en la que se ha realizado la persistencia de datos en la aplicación. Comenzando por un diagrama entidad-relación y como se ha implementado este en la base de datos.

El capítulo 6 precisa la arquitectura de la aplicación y los patrones de diseño utilizados.

El capítulo 7 puntualiza el diseño y la funcionalidad.

Posteriormente se tratan las conclusiones y el trabajo a desarrollar en futuro en el capítulo 8.

En último lugar se encuentra la bibliografía y los anexos donde se incluye una guía de uso de la aplicación y una guía de instalación.

1. Introduction

This first chapter explains the motivation of the developed application, its objectives and describes the memory structure.

1.1. Motivation

Maintaining motivation and interest during the learning process is often complicated, even more so when using conventional methodologies. These ones focus on memorizing concepts and follow a unique itinerary for all students.

The use of game elements in education is increasingly frequent. This methodology gains ground as a means of training since it facilitates learning, increases motivation and participation.

Programming languages are not exempt from being able to use this method, which allows constant feedback. As for learning it is not possible to use a product that fits all the people following the same itinerary regardless of the level or the speed of the students, it would be convenient to be able to generate different paths that are developed taking into account the results obtained during the course.

In this project, a tool has been developed that implements a course management system to learn Python. This system allows teachers to create and manage courses that will form the paths that students will follow. The system makes it easy for students to follow these itineraries.

1.2. Objectives

The objective of this project is to develop a tool that allows generating the courses and creating paths to follow, according to the scores obtained. All this using the aforementioned game mechanics. The specific objectives set were the following:

- Develop a web application that allows teachers to create the courses, questions and itineraries to follow. In addition, to obtain information on the performance of the students enrolled in the courses.

- Implement a REST API that contains the necessary logic to receive all the mobile application requests.

- Develop an Android mobile application through which students can use the tool to complete the courses.

1.3. Memory structure

The memory of this project is structured as follows. Chapter 1 introduces work done and motivation for development. The memory structure is also described.

Tools with functionality similar to that developed in this project are discussed in Chapter 2.

Chapter 3 details the technology used for development.

Chapter 4 specifies the application use cases divided into 3 modules: teachers, students and commons.

Chapter 5 discusses how data persistence has been performed in the application. Starting with an entity-relationship diagram and how this has been implemented in the database.

Chapter 6 details the application architecture and design patterns used.

Chapter 7 spells out design and functionality.

Subsequently, the conclusions and the work to be carried out in the future are discussed in Chapter 8.

Lastly, there is the bibliography and the annexes that include a guide for using the application and an installation guide.

2. Estado del arte

En este capítulo se recogen las características más importantes de algunos productos similares en el mercado actual.

Se analizan productos con características similares. Deben ser aplicaciones que utilicen mecánicas de juego y se encuentren en el marco del M-Learning (Mobile-Learning), es decir, están destinadas al aprendizaje.

El M-learning es una rama del E-learning que aprovecha todas las posibilidades de los dispositivos móviles para aprender sin necesidad de una ubicación o momento específico. En este sentido, la interacción y retroalimentación ocurren de forma inmediata. A continuación se muestran algunos ejemplos de aplicaciones de este tipo.

2.1. Kahoot!

Es una plataforma de aprendizaje basada en el juego que facilita la creación, compartición y resolución de cuestionarios de evaluación. Está disponible en app y versión web.[1]

Una vez registrado en la plataforma, el profesor crea un concurso con cuestionarios donde los alumnos son los concursantes. Además de los cuestionarios creados por el profesor, éste podrá disponer de un repositorio de cuestionarios creados por otros usuarios. Los alumnos podrán acceder al cuestionario a través de un código pin facilitado por el profesor. Una vez acceden, eligen su nombre de usuario y contestan a las distintas preguntas a través de un dispositivo móvil.

Al finalizar se genera un ranking con las puntuaciones obtenidas por los alumnos. Las puntuaciones dependen de la cantidad de respuestas acertadas y la velocidad de respuesta. La aplicación permite exportar los resultados a excel o Google Drive. [2]

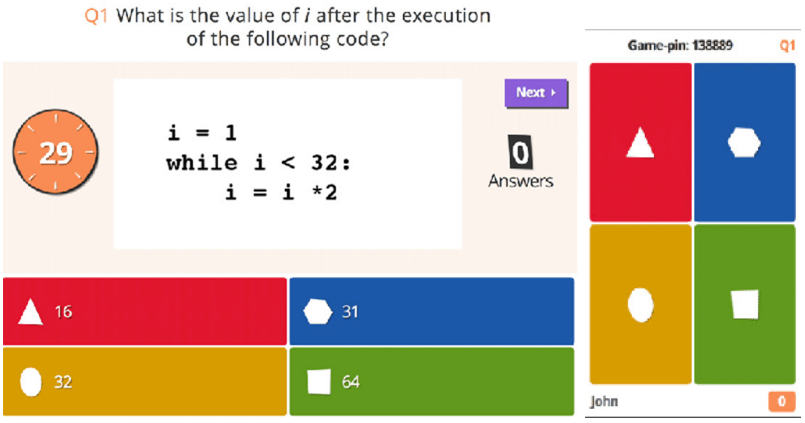


Figura 1. Captura de pantalla de Kahoot!.

2.2. Duolingo

Plataforma destinada al aprendizaje de idiomas utilizando recursos de juegos. Es la aplicación de educación más descargada a nivel mundial. [3] Además de su versión web, la plataforma cuenta con aplicación para Android, IOS y Windows Phone.

Una vez registrado en la aplicación se selecciona el idioma que queremos aprender. A 30 de junio de 2020 la aplicación cuenta con 38 idiomas posibles. Número que aumenta constantemente gracias a su incubadora de idiomas.

Mediante distintas actividades (construir oraciones, emparejar palabras, reconocer sonidos, etc) que se apoyan en imágenes, audios y traducciones se aprende el significado de un gran número de oraciones de forma intuitiva. Los progresos de las distintas unidades se visualizan de forma gráfica que indica el grado de dominio de la unidad. Duolingo registra los contenidos con más errores para ayudar a reforzarlos. [4]

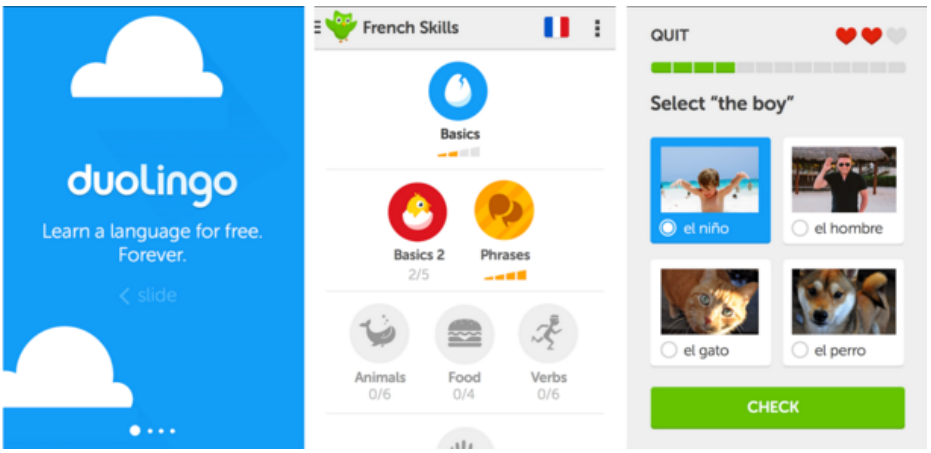


Figura 2. Captura de pantalla de Duolingo.

2.3. Socrative

Herramienta cuya finalidad es el soporte en el aula. Por ello, puede ser utilizado para feedbacks, evaluaciones mediante quiz.[5]

Socrative cuenta con aplicación web y móvil. Su funcionamiento es similar al de las aplicaciones mencionadas anteriormente. Una vez registrado en la aplicación el profesor podrá crear o importar los exámenes (Quiz). Dentro del examen se crean preguntas, a 17 de junio de 2020 cuenta con tres formatos de preguntas, verdadero o falso, opción múltiple y respuesta corta. A estas preguntas se pueden añadir imágenes de apoyo.

El examen creado se comparte con los alumnos quienes contestarán una pregunta a la vez a su propio ritmo. Cuando se han contestado todas el alumno envía las respuestas y obtiene información instantánea del resultado.

La aplicación cuenta con otros modos de uso, como la carrera espacial donde los alumnos trabajan en equipo incentivando la colaboración. También permite al profesor lanzar una pregunta única en un momento puntual sin tener que generar un examen.

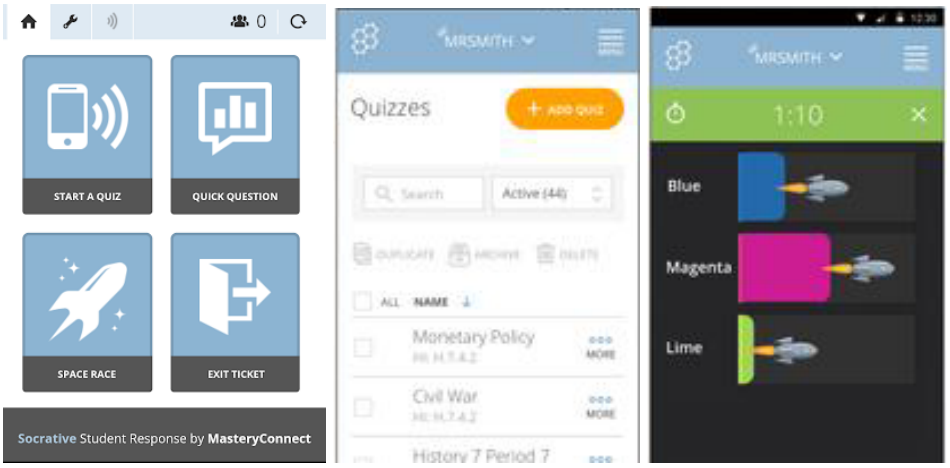


Figura 3. Captura de pantalla de Socrative.

2.4. Trivinet

Es un tivia online creada por un profesor con la idea de utilizar esta herramienta como recurso didáctico que permita a sus alumnos aprender jugando.[6]

Trivinet también cuenta tanto con aplicación web como móvil. Esta cuenta con un único formato de pregunta, opción múltiple. La aplicación permite crear grupos y crear preguntas dentro de los grupos. Según los resultados se genera un ranking de los participantes. Los grupos generados por la comunidad de Triviner pueden ser clonados para utilizar como base las preguntas generadas en el grupo original y ampliarlas según la necesidad del administrador del grupo clonado.

Figura 4. Captura de pantalla de Trivinet.

3. Tecnología empleada

En este capítulo se van a describir las herramientas tecnológicas utilizadas en el desarrollo del proyecto.

3.1. Herramientas del cliente

Para la capa de presentación de la interfaz web se ha utilizado HTML5, CSS3 y Bootstrap 4.

3.1.1. HTML5

HTML (HyperText Markup Language) es un lenguaje de marcado para el desarrollo de páginas web. Define una estructura básica y un código para indicar el contenido de una página web. Esto se genera mediante etiquetas de marcado conocidas como tags.

HTML5 [7] es la última versión de este lenguaje de marcado. Esta versión establece un conjunto de nuevos elementos y atributos con significado semántico que manifiestan las nuevas necesidades de las páginas web modernas. Algunos de estos elementos son las etiquetas *video*, *figure* y *section*. Además de estas etiquetas en esta versión el lenguaje proporciona mecanismos de conexión a servidores de formas nuevas e innovadoras.

3.1.2.3. CSS3

CSS [8] (Cascading Style Sheets) es el lenguaje utilizado para describir la presentación de una página web. Una hoja de estilo CSS se compone de reglas que modifican la manera en que un documento, normalmente HTML, es presentado. Una regla CSS se compone de un selector, que seleccionará los elementos HTML que se actualizarán a través de una serie de propiedades.

Está diseñado principalmente para marcar la separación del contenido, indicado por el código HTML y la presentación de este. Esto hace posible que un mismo documento HTML se pueda presentar con estilos distintos para diferentes métodos de renderizado como puede el tamaño de pantalla.

CSS funciona con un esquema prioritario que determina qué reglas se aplican si más de una coincide para un elemento en particular. Estas reglas se aplican mediante un sistema de cascada, de modo que las prioridades son calculadas y asignadas a las reglas, así que los resultados son predecibles.

3.1.3. Bootstrap 4

Bootstrap 4 [9] es una biblioteca multiplataforma o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web. Contiene plantillas de diseño

con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como extensiones de JavaScript adicionales. A diferencia de muchos frameworks web, solo se ocupa del desarrollo front-end.

Una de las características claves de Bootstrap es el uso de un sistema de grid de doce columnas donde se inserta el contenido. Esta característica es fundamental para que los diseños web sean adaptables (responsive) entre distintos dispositivos con distintos tamaños de pantalla.

Bootstrap 4 es la última versión de este framework. Algunas de las características principales de esta versión son el nuevo sistema de grid para pantallas más pequeñas, nuevas plantillas de elementos HTML y CSS.

3.2. Herramientas del servidor

3.2.1. Node.JS

Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación JavaScript. Ideado como un entorno de ejecución de JavaScript orientado a eventos asíncronos, Node.js está diseñado para crear aplicaciones network escalables. Esto contrasta con el modelo de concurrencia más común de hoy en día, en el que se emplean hilos del Sistema Operativo [10].

Node.js permite gestionar un desarrollo homogéneo entre cliente y servidor. Al ser combinado con una base de datos documental permite trabajar en un entorno de desarrollo JavaScript unificado.

Las interfaces de programación de aplicaciones (API) de Node.js son asíncronas, esto significa que los servidores Node no esperan la respuesta de una API. El servidor continúa con la siguiente API después de hacer la llamada y el sistema de notificaciones de eventos de node ayuda al servidor a dar la respuesta a la llamada API anterior.

Node.js se encuentra bajo licencia MIT.

3.2.2. NPM

Es el sistema de gestión de paquetes por defecto para Node.js [11]. Consiste en una interfaz de línea de comandos y una base de datos online con paquetes públicos y privados.

NPM consiste en tres componentes distintos. La web, desde donde se encuentran los paquetes, se estructuran organizaciones para administrar acceso público o privado a estos paquetes. La interfaz de línea de comandos (CLI) es el medio que tienen los desarrolladores para interactuar con NPM. Y el registro es una base de datos pública de software Javascript y metainformación del mismo.

3.2.3. Express

También conocido como Express.js es una infraestructura de aplicaciones web Node.js. Diseñado para construir aplicaciones web y APIs [12].

Express permite construir aplicaciones web en un periodo corto de tiempo ya que brinda mecanismos para facilitar esta construcción. Proporciona un enrutamiento simple para las solicitudes realizadas por los clientes. También proporciona un middleware que se encarga de tomar decisiones para dar las respuestas correctas a estas solicitudes.

3.2.4. EJS

Lenguaje de plantillas que permite generar páginas HTML utilizando JavaScript [13]. Mediante una sintaxis sencilla de etiquetas EJS combinado con HTML podrá cargar contenido de forma dinámica en la página. Proporciona además mecanismos de control de flujo.

EJS compila y renderiza de forma muy rápida. Esto es posible gracias a la caché estática de JavaScript que evita que haya que compilar la página cada vez que se renderiza.

3.2.5. Bcrypt

Es una librería que permite aplicar funciones hash a las contraseñas para almacenar estas de forma segura [14].

Bcrypt provee mecanismos sencillos para almacenar nuestras contraseñas de forma segura y no en texto plano. Además utiliza técnicas de criptografía como la extensión de clave.

3.2.6. Retrofit

Es un cliente REST para Android y Java [15]. Permite hacer peticiones GET, POST, PUT, PATCH, DELETE y HEAD además de gestionar diferentes tipos de parámetros y parsear la respuesta a un Objeto Java de forma automática.

Mediante Retrofit y una vez definido el modelo de clases en formato JSON y la interfaz de operaciones HTTP se pueden hacer peticiones a un servidor desde un dispositivo Android.

3.3. Persistencia de la información

3.3.1. MongoDB

Es un sistema de base de datos NoSQL, orientado a documentos y de código abierto.En lugar de guardar los datos en tablas, tal y como se hace en las bases de

datos relacionales, MongoDB guarda estructuras de datos BSON (una especificación similar a JSON) con un esquema dinámico, haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida [16]. En todos los documentos se almacenan campos generados por MongoDB

- _*id*: Identificador único del documento generado por MongoDB.
- _v: Campo de versión del documento. Contiene la revisión interna del documento. Arquitectura de la aplicación

3.4. Otras herramientas

3.4.1. Android Studio

Entorno de desarrollo integrado oficial para la plataforma Android [17]. Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece incluso más funciones que aumentan tu productividad cuando desarrollas apps para Android, como las siguientes [18]:

- Un sistema de compilación flexible basado en Gradle
- Un emulador rápido y cargado de funciones
- Un entorno unificado donde puedes desarrollar para todos los dispositivos Android
- Aplicación de cambios para insertar cambios de códigos y recursos a la aplicación en ejecución sin reiniciar la aplicación
- Variedad de marcos de trabajo y herramientas de prueba
- Herramientas de Lint para identificar problemas de rendimiento, usabilidad y compatibilidad de la versión.

3.4.2. Git

Sistema de control de versiones utilizado[19]. Es una herramienta útil cuando los requisitos del proyecto cambian, en ocasiones será necesario volver a versiones anteriores del código. También permite explorar distintas ramas de desarrollo de forma independiente entre sí que pueden ser fusionadas posteriormente.

3.4.3. Visual Studio Code

Visual Studio Code [20] ha sido el editor de código utilizado para la parte web del proyecto. Incluye soporte para la depuración, control de versiones y resaltado de sintaxis.Incluye soporte incorporado para JavaScript y Node.js

4. Casos de uso

En este capítulo se describen los casos de uso definidos para la aplicación desarrollada en este trabajo de fin de grado. Los casos de uso se agrupan de acuerdo a la funcionalidad asociada a cada actor.

4.1. Actores de la aplicación

Los actores definidos para los casos de uso que se muestran son:

- Profesores: Serán los encargados de crear, editar o eliminar los cursos de la aplicación y generar los caminos. Además podrán visualizar información de todos los alumnos

- Alumnos: Los alumnos podrán realizar los cursos y visualizar la información de sus resultados. No podrán editar el contenido de los cursos ni visualizar resultados de otros alumnos.

4.2. Casos de uso asociados al alumno

A continuación se van a mostrar los diferentes casos de uso definidos para el alumno y que se muestran en el diagrama de la figura 5.

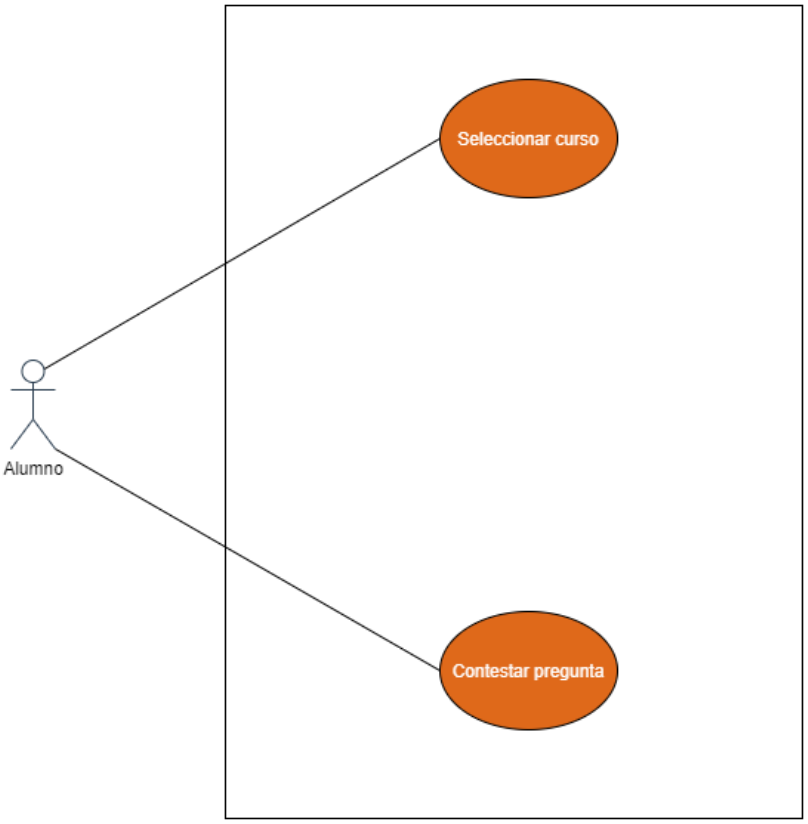


Figura 5. Casos de uso definidos para el actor alumno

CU-A01	Seleccionar curso
Objetivos asociados	Ejecuta la actividad correspondiente de un curso
Entradas	id_curso
Salidas	Confirmación de inicio de actividad
Precondición	● El usuario debe haber iniciado sesión ● El curso debe existir.
Secuencia normal	1. Muestra la pantalla principal. 2. El usuario selecciona el curso a ejecutar. 3. El sistema redirige al usuario a la pantalla del curso seleccionado mostrando la la pregunta correspondiente.
Postcondición	La actividad se debe estar ejecutando

CU-A02	Contestar pregunta
Objetivos asociados	Contesta una pregunta de la actividad en ejecución.
Entradas	id_pregunta
Salidas	Mensaje de realimentación
Precondición	● El usuario debe haber iniciado sesión ● Se debe estar ejecutando una actividad
Secuencia normal	1. Muestra la pantalla de la pregunta. 2. El usuario selecciona la respuesta 3. El sistema redirige al usuario a la siguiente pregunta.
Postcondición	Se ha contestado la pregunta

4.3. Casos de uso asociados al profesor.

A continuación se van a mostrar los diferentes casos de uso definidos para el profesor. En la figura 6 se muestra el diagrama de casos de uso asociado al actor profesor.

Figura 6. Casos de uso definidos para el actor profesor

CU-P01	Crear curso
Objetivos asociados	Añade un curso a la aplicación
Entradas	nombre_curso, descripcion_curso, nivel_curso, debloqueado_curso, aDesbloquea_curso, bDesbloquea_curso
Salidas	Mensaje de realimentación
Precondición	● El usuario debe haber iniciado sesión ● El usuario debe ser un profesor
Secuencia normal	1. Muestra la pantalla de añadir curso. 2. El usuario introduce los datos requeridos 3. El sistema redirige al usuario a la pantalla de cursos.
Postcondición	Se ha añadido el curso

CU-P02	Editar curso
Objetivos asociados	Edita un curso
Entradas	nombre_curso, descripcion_curso, nivel_curso, debloqueado_curso, aDesbloquea_curso, bDesbloquea_curso
Salidas	Mensaje de realimentación
Precondición	● El usuario debe haber iniciado sesión ● El usuario debe ser un profesor ● El curso debe existir.
Secuencia normal	1. Muestra la pantalla de editar curso 2. El usuario introduce los datos requeridos 3. El sistema redirige al usuario a la pantalla de cursos.
Postcondición	Se ha editado el curso.

CU-P03	Eliminar curso
Objetivos asociados	Elimina un curso
Entradas	id_curso
Salidas	Mensaje de realimentación
Precondición	● El usuario debe haber iniciado sesión ● El usuario debe ser un profesor ● El curso debe existir.
Secuencia normal	1. Muestra la pantalla del curso. 2. El usuario selecciona el eliminar curso 3. El sistema redirige al usuario a la pantalla de cursos.
Postcondición	Se ha eliminado el curso.

CU-P04	Añadir pregunta
Objetivos asociados	Añade una pregunta a un curso
Entradas	nombre_pregunta, opciones_pregunta
Salidas	Mensaje de realimentación
Precondición	● El usuario debe haber iniciado sesión ● El usuario debe ser un profesor ● El curso debe existir
Secuencia normal	1. Muestra la pantalla de añadir preguntas. 2. El usuario introduce los datos requeridos 3. El sistema redirige al usuario a la pantalla de preguntas
Postcondición	Se ha añadido la pregunta al curso

CU-P05	Editar pregunta
Objetivos asociados	Edita una pregunta de un curso
Entradas	nombre_pregunta, opciones_pregunta
Salidas	Mensaje de realimentación
Precondición	● El usuario debe haber iniciado sesión ● El usuario debe ser un profesor ● La pregunta debe existir
Secuencia normal	1. Muestra la pantalla de editar preguntas. 2. El usuario introduce los datos requeridos 3. El sistema redirige al usuario a la pantalla de preguntas.
Postcondición	Se ha editado la pregunta

CU-P06	Eliminar pregunta
Objetivos asociados	Elimina una pregunta de un curso
Entradas	nombre_pregunta, opciones_pregunta
Salidas	Mensaje de realimentación
Precondición	● El usuario debe haber iniciado sesión ● El usuario debe ser un profesor ● La pregunta debe existir
Secuencia normal	1. Muestra la pantalla de eliminar preguntas. 2. El usuario confirma que desea eliminar la pregunta 3. El sistema redirige al usuario a la pantalla de preguntas.
Postcondición	Se ha eliminado la pregunta.

4.4. Casos de uso comunes a varios actores.

A continuación se van a mostrar los diferentes casos de uso comunes para todos los usuarios. En la figura 7 se muestra el diagrama de casos de uso asociado al actor profesor.

Figura 7. Casos de uso comunes

CU-C01	Inicio de sesión
Objetivos asociados	Inicia la sesión de un usuario en la aplicación.
Entradas	email_usuario, contraseña_usuario.
Salidas	Confirmación de inicio de sesión
Precondición	● El usuario debe existir en la base de datos. ● La combinación de email y contraseña deben ser válidos y existir en la BBDD
Secuencia normal	1. Muestra la pantalla de Inicio de sesión. 2. El usuario introduce sus credenciales. 3. El sistema redirige al usuario a la pantalla principal.
Postcondición	La sesión debe quedar iniciada
Excepciones	E1 - El mail y/o contraseña son erróneos

CU-C02	Cierre de sesión
Objetivos asociados	Cierra la sesión de un usuario en la aplicación.
Entradas	
Salidas	Confirmación de cierre de sesión
Precondición	● La sesión debe estar iniciada
Secuencia normal	1. El usuario hace clic en el botón de cierre de sesión 2. Se cierra la sesión. 3. Se redirige a la pantalla de inicio de sesión
Postcondición	La sesión debe quedar cerrada
Excepciones	

CU-C03	Crear cuenta
Objetivos asociados	Registrar a un usuario en el sistema
Entradas	email_usuario, contraseña_usuario
Salidas	Usuario creado
Precondición	● La dirección de correo debe ser válida ● La contraseña debe tener más de 6 caracteres alfanuméricos ● La repetición de contraseña debe coincidir
Secuencia normal	1. El usuario accede a la página de registro 2. Introduce los datos requeridos 3. Envía el formulario 4. Se redirige a la página principal
Postcondición	La sesión debe quedar iniciada
Excepciones	E1- El nombre de usuario ya existe E2- La contraseña no es válida o no coincide con su repetición

CU-C04	Editar cuenta
Objetivos asociados	Modifica los datos de un usuario
Entradas	id_usuario
Salidas	Usuario modificado
Precondición	● El usuario debe haber iniciado sesión
Secuencia normal	1. Muestra la pantalla de modificar datos 2. El usuario modifica los datos 3. Se envía el formulario 4. Se redirige a la página principal
Postcondición	Los datos se han modificado

CU-C05	Eliminar cuenta
Objetivos asociados	Dar de baja a un usuario del sistema
Entradas	id_usuario
Salidas	Usuario dado de baja
Precondición	● El usuario debe existir ● La sesión debe estar iniciada
Secuencia normal	1. El usuario accede a la página de opciones 2. Solicita la baja de la cuenta 3. Vuelve a introducir la contraseña 4. Se redirige a la página de inicio de sesión
Postcondición	La sesión debe quedar cerrada y el usuario eliminado
Excepciones	E1- La contraseña introducida es incorrecta

CU-C06	Ver tabla de puntuación
Objetivos asociados	Muestra una lista con los usuarios ordenados por puntuación.
Entradas	
Salidas	Lista de puntuación.
Precondición	● La sesión debe estar iniciada
Secuencia normal	1. El usuario accede a la página principal 2. Selecciona la opción de ver lista de puntuación 3. Se redirige a la página de lista de puntuación
Postcondición	Se ha mostrado la tabla
Excepciones	

5. Modelo de datos

En este capítulo se describe el modelo de datos definido para realizar la persistencia de la información gestionada por la herramienta. En la sección 1 se describe el modelo E-R del sistema y en la sección 2 se mostrará su implementación usando una base de datos NoSQL.

5.1. Modelo ER

En la figura 8 se muestra el diagrama entidad-relación utilizado para modelar los datos de la aplicación. En este se presentan las entidades *Usuario*, *Curso, Completados y Pregunta* y las relaciones entre las mismas. Donde un usuario (alumno o profesor) podrá tener completados varios cursos y a su vez cada curso tendrá varios usuarios que lo hayan completado. Los profesores que lo crean, podrán editar o eliminar, y los alumnos únicamente para cumplimentar. Cada curso tendrá distintas preguntas.

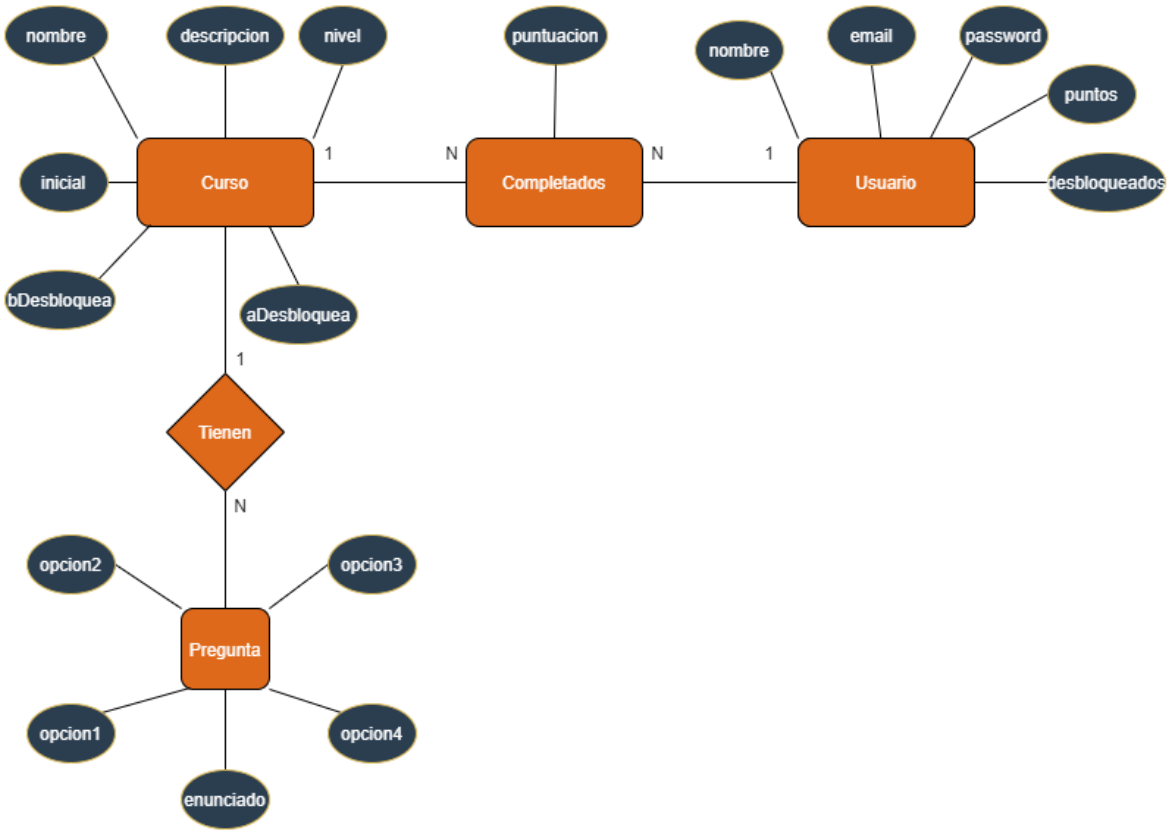


Figura 8. Diagrama entidad-relación.

5.2. Implementación de la base de datos.

Para implementar el modelo E-R definido en la sección anterior se ha utilizado MongoDB. En la figura 9 se muestran las colecciones definidas. A continuación se procede a describir estas colecciones.

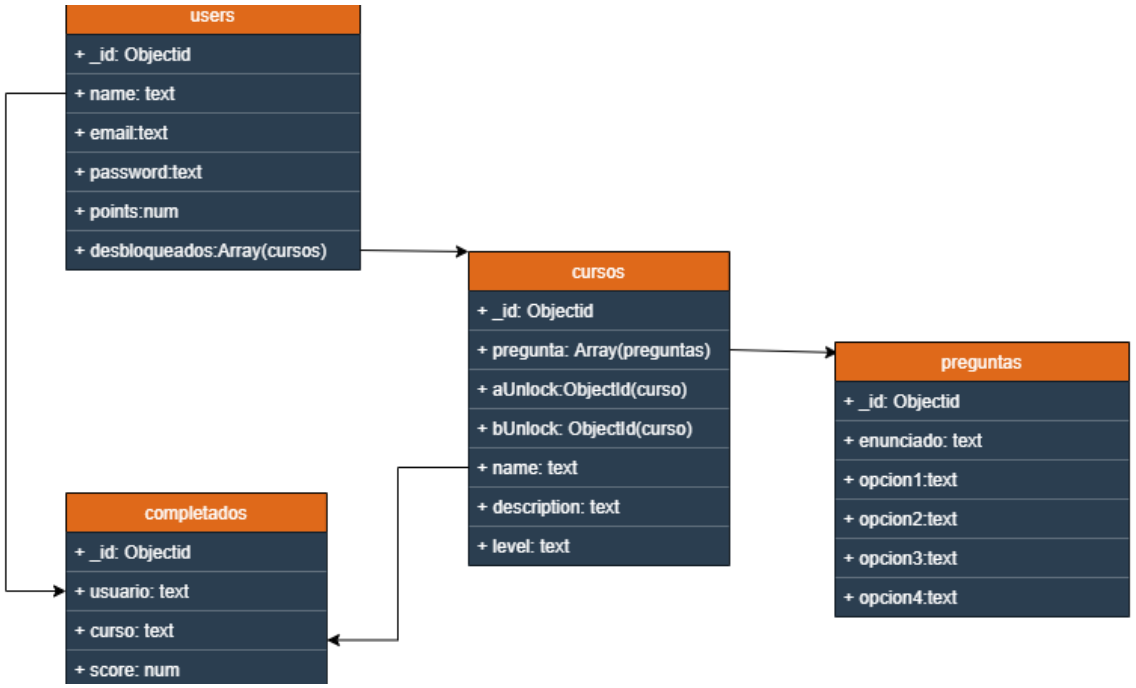


Figura 9. Colecciones de la base datos.

5.2.1. Users

En la figura 10 se muestra un ejemplo de documento visto desde MongoDB Compass.. Esta colección almacena la información de los usuarios. En la figura 4 se muestra un ejemplo de documento visto desde MongoDB Compass. Los campos que almacena esta colección son los siguientes:

- *name*: Nombre del usuario.
- *email*: Email del usuario.
- *password*: Contraseña del usuario codificada mediante una función hash
- *points*: Puntos totales del usuario, que va consiguiendo al completar cursos. Esta propiedad solo la tendrán los alumnos.
- *desbloqueados*: Array de referencias a cursos. Aquí se almacenan los cursos a los que el usuario tiene acceso.

Figura 10. Documento de la colección usuario en MongoDB Compass.

5.2.2. Cursos

Se procede a detallar la información relativa a los cursos. En la figura 11 se muestra un ejemplo de documento visto desde MongoDB Compass. Los campos que almacena esta colección son los siguientes:

- *name*: Nombre del curso.
- *description*: Descripción corta del curso.
- *level*: Nivel del curso. Se puede seleccionar entre las opciones principiante, Intermedio o avanzado.
- *aUnlock*: Guarda una referencia al curso que se desbloquea si se aprueba este.
- *bUnlock*: Guarda una referencia al curso que se desbloquea si se suspende este.
- *unlocked*: Booleano que dice si este campo estará desbloqueado desde su creación sin necesidad de que se desbloquee aprobando otro curso.
- *pregunta*: Array que almacena las referencias a las preguntas del curso.

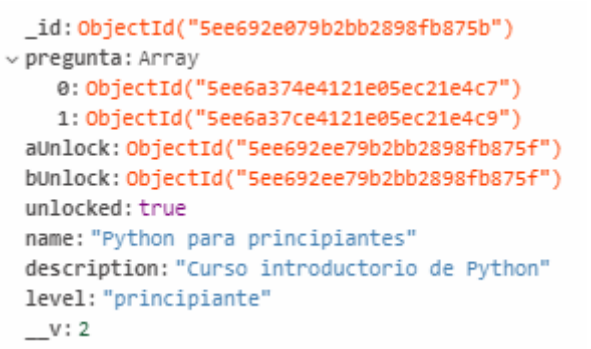


Figura 11. Documento de la colección curso en MongoDB Compass.

5.2.3. Pregunta

A continuación se especifica la información almacenada en los documentos pregunta. En la figura 12 se muestra un ejemplo de documento visto desde MongoDB Compass. Los campos que almacena esta colección son los siguientes:

- *enunciado*: Enunciado de la pregunta.
- *opción 1*: Primera opción de respuesta, en este se almacenará la respuesta correcta.
- *opción2*: Segunda opción de respuesta.
- *opción3* : Tercera opción de respuesta.
- *opción4* : Cuarta opción de respuesta.

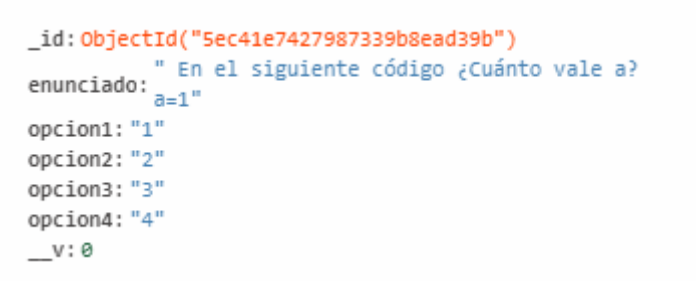


Figura 12. Documento de la colección pregunta en MongoDB Compass.

5.2.4. Completados

La colección completados almacena la puntuación obtenida por cada usuario al completar cada curso. En la figura 13 se muestra un ejemplo de documento visto desde MongoDB Compass.

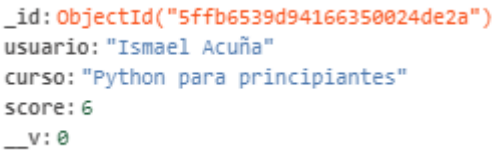


Figura 13. Documento de la colección completados en MongoDB Compass.

5.2.5. Sessions

Además de las entidades descritas en el diagrama entidad relación se utiliza una colección auxiliar para almacenar los datos relativos a las sesiones de los usuarios. En la figura 14 se muestra un documento de ejemplo.

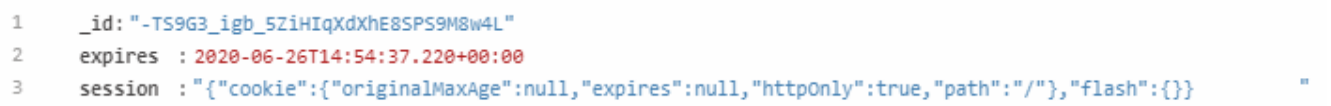


Figura 14. Documento de la colección sessions en MongoDB Compass.

6. Arquitectura de la aplicación

En este capítulo se describe la arquitectura de la aplicación. En la primera sección se explica el modelo cliente-servidor y como se ha utilizado en el desarrollo de la aplicación, y en la sección 2 se describen los patrones de diseño empleados.

Para desarrollar la aplicación se ha empleado un modelo cliente-servidor que se caracteriza porque las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, quien le da respuesta [20].

En la figura 8 se muestra el diseño de la arquitectura de la aplicación. El navegador y/o la aplicación móvil ejecutan la parte cliente de la aplicación realizando peticiones HTTP a los servicios que expone la API REST, la cual de forma oculta para el cliente procesa las peticiones, obtiene los datos necesarios de la base de datos y envía la respuesta de vuelta al cliente.

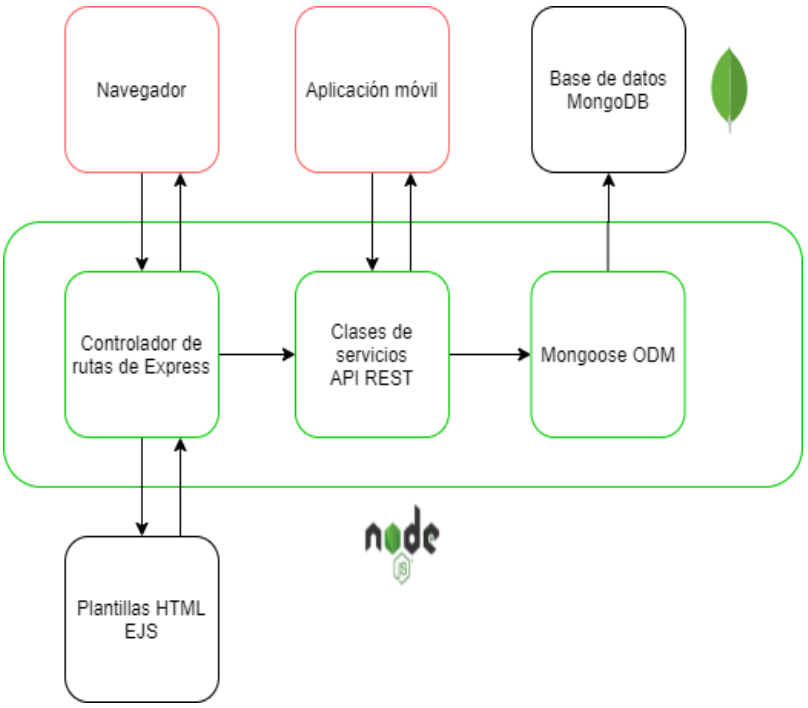


Figura 15. Esquema de la arquitectura del sistema.

El MVC (Modelo Vista controlador) es un patrón de arquitectura de software, que separa los datos y principalmente lo que es la lógica de negocio de una aplicación de su representación y el módulo encargado de gestionar los eventos y las comunicaciones [21].

Se ha aplicado este patrón dentro del marco de Express, donde el modelo es lo que manejará los datos de la aplicación, almacenados en MongoDB. Además Mongoose nos facilita las operaciones con estos datos.

La vista generará información en la pantalla. Express nos proporciona métodos para encapsular las respuestas y enviarlas a las vistas.

La función de controlador la hacen las clases del middleware de rutas. Que obtienen la información de las vistas, solicitan los datos y encapsulan y devuelven las respuestas.

7. Diseño de la aplicación

En este capítulo se describirán aspectos acerca del diseño y la implementación de las funcionalidades más significativas del proyecto, tanto de la aplicación web, la API implementada y la aplicación móvil.

El propósito ha sido conseguir un diseño simple y atractivo. Que a la vez fuese coherente entre las distintas plataformas.

7.1. Colores y tipografías

Como se ha comentado con anterioridad para facilitar el desarrollo de las interfaces de usuario se ha utilizado Bootstrap

Para definir los colores y tipografías se ha utilizado un tema desarrollado para bootstrap que se encuentra bajo licencia MIT [22]. En la figura 16 se muestra el código de referencia a estas bibliotecas y temas.

```
7   <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
8   <link rel="stylesheet" href="https://bootswatch.com/4/superhero/bootstrap.min.css"> <!--Tema Bootstrap-->
9   <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.6.3/css/all.css"
```

Figura 16. Código de referencias CSS.

Este tema utiliza como tipografía principal la fuente ‘Lato’

```
--font-family-sans-serif: "Lato", -apple-system, BlinkMacSystemFont,
 "Segoe UI",
 Roboto, "Helvetica Neue", Arial, sans-serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol";
--font-family-monospace: SFMono-Regular, Menlo, Monaco, Consolas, "Liberation Mono", "Courier New", monospace;
```

Figura 17. Tipografía utilizadas en la aplicación.

El tema también define varios colores que se muestran a continuación.

```
--blue: #df691a;
--indigo: #6610f2;
--purple: #6f42c1;
--pink: #e83e8c;
--red: #d9534f;
--orange: #f0ad4e;
--yellow: #f0ad4e;
--green: #5cb85c;
--teal: #20c997;
--cyan: #5bc0de;
--white: #fff;
--gray: #868e96;
--gray-dark: #343a40;
--primary: #df691a;
--secondary: #4e5d6c;
--success: #5cb85c;
--info: #5bc0de;
--warning: #f0ad4e;
--danger: #d9534f;
--light: #abb6c2;
--dark: #4e5d6c;
```

Figura 18. Colores de la aplicación.

7.2. Funcionalidad de la aplicación web

En este apartado se procederá a describir las funcionalidades implementadas. Como el sistema tiene dos modos de uso separados, aplicación web y aplicación móvil, se describirán por separado.

7.2.1. Funcionalidades web

Las funcionalidades de la web están relacionadas directamente con los casos de uso asociados a los profesores.

7.2.1.1. Gestión de cursos

Cuando un profesor se autentica en el sistema, accede a la pantalla principal o *dashboard* (Figura 19) en la que se muestran los cursos creados. Así mismo puede crear nuevos cursos, ver el ranking de usuarios, acceder a las opciones de la aplicación o cerrar sesión.

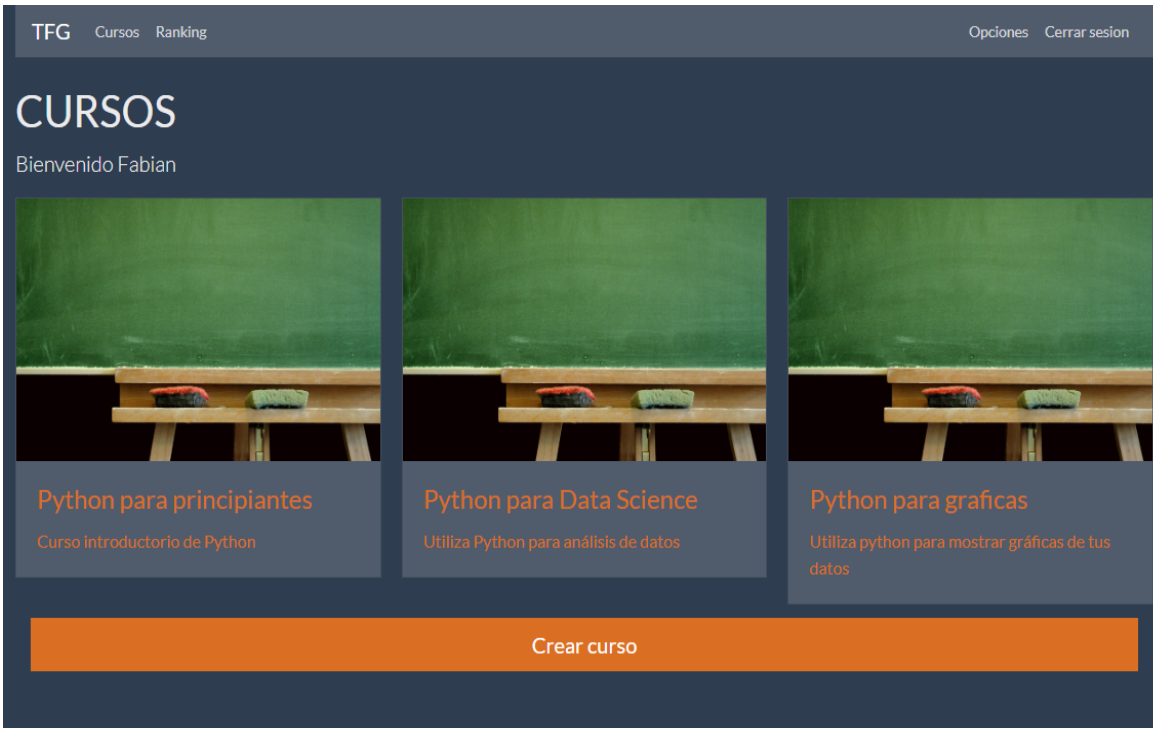


Figura 19. Pantalla principal.

Para mostrar todos los cursos se aprovechan las ventajas de EJS (capítulo 3.2.4) y poder ejecutar el código desde la plantilla HTML (figura 20). La variable curso en este caso se define en el controlador, evitando el error de acceder al modelo desde las vistas (figura 21).

```
<div class="card-columns">
  <% cursos.forEach(function(cursos){%>
  <div class="card">
    <img class="card-img-top img-fluid" src="/img/cursoheader.jpg" alt="Card image cap">
    <div class="card-body">
      <a href="/curso/?id=<%= cursos.id %>">
      <h4 class="card-title"> <%= cursos.name %></h4>
      <p class="card-text"><%=cursos.description %></p>
    </a>
    </div>
  </div>
 <% })%>
</div>
```

Figura 20. Código para mostrar cursos en el dashboard.

Se puede observar en el código que estos cursos contienen enlaces a la página del curso.

```
//DASHBOARD
router.get ('/dashboard',ensureAuthenticated, (req, res)=>
Curso.find({}, function(err, cursos) {
 res.render('dashboard',{
     name: req.user.name,
     cursos:cursos
    });
}));
```

Figura 21. Variable cursos en el controlador.

7.2.1.2. Creación de un curso

Una tarea fundamental de la aplicación es la creación de cursos. Para crear un curso, desde la pantalla principal (Figura 19) se accede mediante el botón ‘Crear curso’ al formulario de creación de cursos (Figura 22).

Figura 22. Formulario de creación de cursos.

En este formulario (Figura 22) se definen los datos del curso. Es importante destacar que mediante la elección de qué cursos se desbloquean dependiendo de la puntuación obtenida en el curso, se generan distintos caminos posibles para los alumnos. Estos campos nos dan la posibilidad de escoger dentro de una lista con todos los cursos creados (Figura 23).

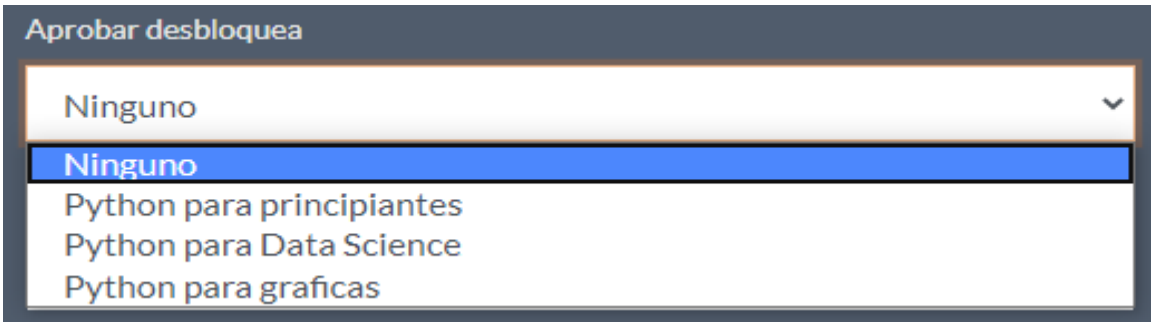


Figura 23. Campo para generar caminos.

Del mismo modo que mostrabamos los cursos en el dashboard, hacemos uso de EJS para poder mostrar esta lista de cursos (ver Figura 24).

```
38          <div class="form-group">
39            <label for="aUnlock">Aprobar desbloquea</label>
40              <select class="form-control" id="aUnlock" name="aUnlock">
41                <option value="none">Ninguno</option>
42                <% cursos.forEach(function(cursos){%>
43                <option value="<%= cursos._id %>"><%= cursos.name %></option>
44                <% })%>
45              </select>
46          </div>
```

Figura 24. Código lista de cursos en campos de desbloquear.

El cuadro o *checkbox* “Desbloqueado de inicio” da la posibilidad que el curso que se está creando aparezca desbloqueado para todos los usuarios. Así se generan los inicios de caminos.

```
79              if(unlocked==="true"){
80                  User.find({})
81                  .then(us=>{
82                      us.forEach(usr => {
83                          if(usr){
84                              usr.desbloqueados.push(newCurso);
85                              usr.save();
86                          }
87                      });
88                  });
89              }
```

Figura 25. Código desbloquear un curso de inicio.

7.2.1.3. Ver ranking

Otra de las opciones a la que se puede acceder desde la pantalla principal o *dashboard* (Figura 19) es la de visualizar el ranking de alumnos (Figura 26). Esta opción muestra un listado con los alumnos ordenados por las puntuaciones obtenidas en los cursos.

RANKING

Posicion	Nombre	Puntos
1	Ismael Acuña	160
2	Tamara Verdu	91
3	Ricard Martinez	61

Figura 26. Ranking de usuarios.

Para obtener este ranking se consultan los usuarios con cero o más puntos (Figura 27). Los documentos de profesores no tienen este atributo y por esa razón no se muestran en el ranking.

```
//RANKING
router.get ('/ranking',ensureAuthenticated, (req, res)=>
User.find({points:{ $gte: 0 }},null,{sort:{points:'descending'}} , function(err, user) {
res.render('ranking',{
    user:user
});
}));
```

Figura 27. Código ranking de usuarios.

Esta consulta devuelve una lista de usuarios que posteriormente se muestra en la página con un formato de tabla (Figura 28).

```
<%- include('./partials/navbar.ejs'); %>
<h1 class="mt-4">RANKING</h1>

<table class="table table-striped table-dark">
    <thead>
        <tr>
            <th>Posicion</th>
            <th>Nombre</th>
            <th>Puntos</th>
        </tr>
    </thead>
    <tbody>
        <% var i=1 ;%>
        <% user.forEach(function(user){ %>
        <tr>
            <td><%= i%></td>
            <td><a href="/profile/?id=<%= user.id %>" ><%= user.name %></td></a>
            <td><%= user.points %></td>
        </tr>
        <%  i++; %>
        <% }) %>
    </tbody>
</table>
```

Figura 28. Código tabla de ranking.

7.2.1.4. Ver perfil de usuario.

Desde el ranking de usuarios el profesor puede acceder al perfil de cada usuario para ver más detalles del mismo.

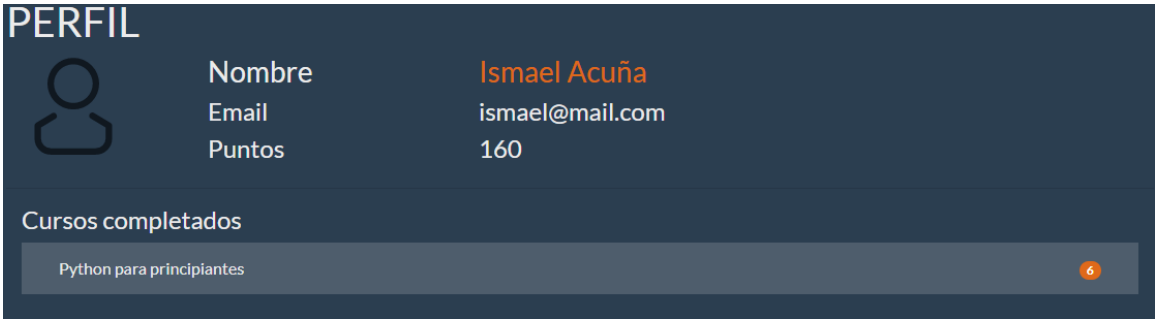


Figura 29. Perfil del usuario.

El perfil de cada usuario muestra información de este como el nombre, email y la puntuación total. Esta información se complementa con una lista de todos los cursos que ha completado y la puntuación obtenida en cada uno.

```
14  router.get ('/*',ensureAuthenticated,(req, res)=>
15      User.findOne({_id:req.query.id}, function(err, user) {
16          Completados.find({usuario:user.name},function(err, completados){
17              res.render('userProfile',{
18              user:user,
19              completados:completados
20             });
21          })
22  }));
```

Figura 30. Código para obtener cursos completados de un alumno.

Para mostrar esta lista se obtiene la información de la colección Completados como se muestra en la figura 30. Posteriormente se recorre el array obtenido y se imprime a modo de lista (figura 31).

```
32  <div class="container">
33      <ul class="list-group">
34          <% completados.forEach(function(completados){%>
35              <li class="list-group-item d-flex justify-content-between align-items-center"><div class="container">
36                  <div class="float-left">
37                      <%= completados.curso %>
38                  </div>
39                  <div class="float-right">
40                      <span class="badge badge-primary badge-pill"><%= completados.score %> </span>
41                  </div>
42              </div></li>
43          <% })%>
44      </ul>
45  </div>
```

Figura 31. Código para mostrar cursos completados de un alumno.

7.2.1.4. Opciones de la aplicación

Otro acceso que es posible desde la pantalla principal (Figura 19) es a las opciones de la aplicación. En la pantalla de opciones (Figura 32) se puede editar o eliminar el usuario actual.

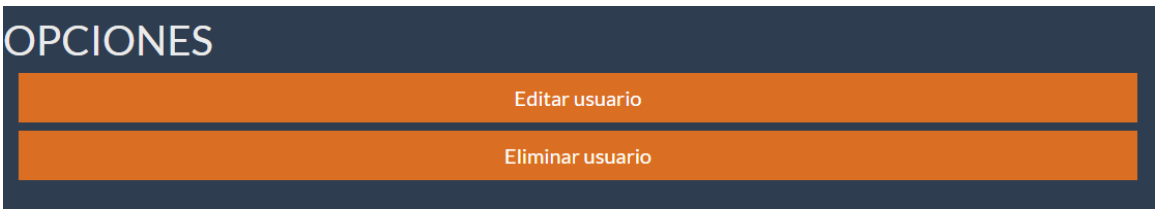


Figura 32. Pantalla de opciones.

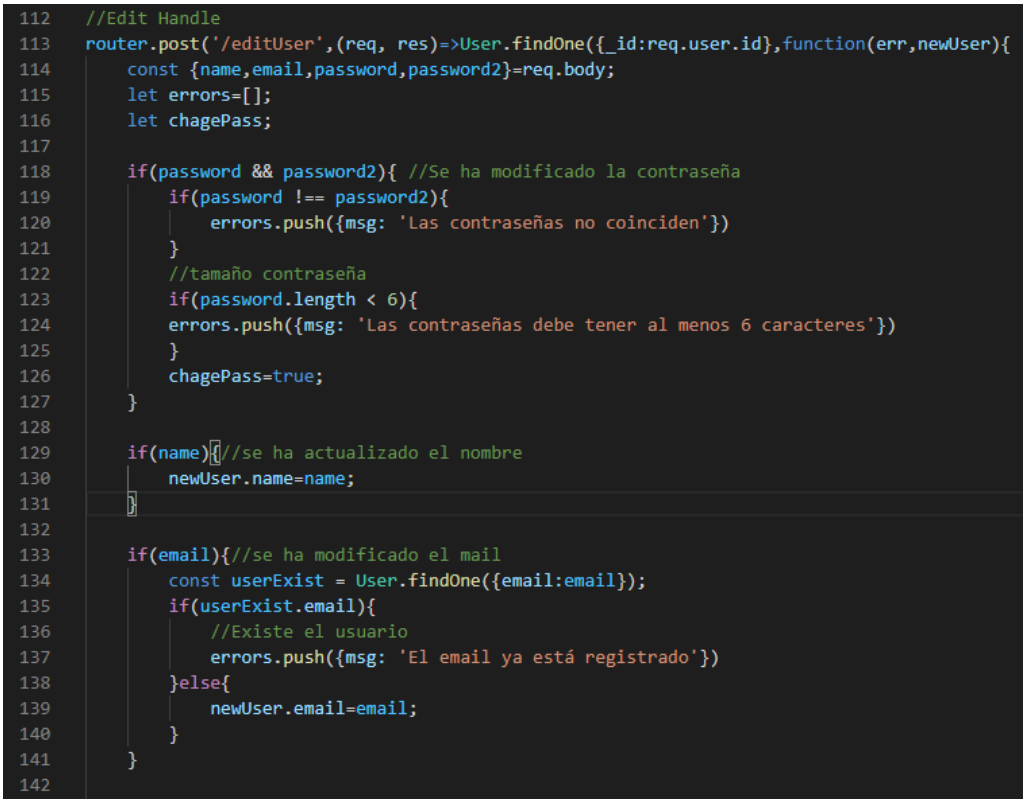
7.2.1.5. Editar usuario

Al acceder a editar usuario desde la pantalla de opciones (Figura 32) aparecerá un formulario similar al de creación de cuenta (Figura 48). En este formulario de edición

(Figura 33) aparecen a modo de marcador de posición (placeholder) algunos datos del usuario actual. No es necesario rellenar el formulario completo, solo los datos a modificar.

Figura 33. Pantalla de edición de usuario.

Una vez modificado los datos y pulsado editar cuenta, se comprueba si se ha insertado algún valor en el formulario, se validan los datos insertados y se guardan únicamente los campos que se han modificado (Figuras 34 y 35).



```
//Edit Handle
router.post('/editUser',(req, res)=>User.findOne({_id:req.user.id},function(err,newUser){
    const {name,email,password,password2}=req.body;
    let errors=[];
    let chagePass;

    if(password && password2){ //Se ha modificado la contraseña
        if(password !== password2){
            errors.push({msg: 'Las contraseñas no coinciden'})
        }
        //tamaño contraseña
        if(password.length < 6){
        errors.push({msg: 'Las contraseñas debe tener al menos 6 caracteres'})
        }
        chagePass=true;
    }

    if(name){//se ha actualizado el nombre
        newUser.name=name;
    }

    if(email){//se ha modificado el mail
        const userExist = User.findOne({email:email});
        if(userExist.email){
            //Existe el usuario
            errors.push({msg: 'El email ya está registrado'})
        }else{
            newUser.email=email;
        }
    }

```

Figura 34. Código de edición de usuario 1.

```
    if(errors.length >0){
        res.render('edit',{
            errors,
            user:newUser
        });
    }else{
        if(chagePass){
            bcrypt.genSalt(10,(error, salt)=>
            bcrypt.hash(req.body.password,salt, (error, hash)=>{
                console.log("HASH: " + hash);
                if(error) throw err;
                //Convertir contraseña en hash
                newUser.password=hash;
                newUser.save();
            }));
        }else{
        newUser.save();
        res.redirect('/options');
    }
    }
}));
```

Figura 35. Código de edición de usuario 2.

7.2.1.6. Eliminar usuario

Además de editar usuario, desde la pantalla de opciones (Figura 32) se puede eliminar el usuario actual. Para eliminar el usuario se busca por id el usuario que está

autenticado en la aplicación y se elimina y se cierra la sesión de la aplicación (Figura 36).

```
25  //DELETE USER
26  router.get ('/deleteUser',ensureAuthenticated, (req, res)=> res.render('delete'));
27  router.get ('/deleteTrue',ensureAuthenticated, function(req, res){
28      User.findByIdAndDelete(req.user._id,function(err){
29          if(err) res.render('delete',{
30              errors
31          })
32          res.redirect('/users/logout');
33      })
34  });
35
```

Figura 36. Código de eliminar usuario.

7.2.1.7. Gestión de curso

La opción a la que es posible acceder desde la pantalla principal (Figura 19) es a un curso. Cuando se accede a un curso se muestra la pantalla de gestión del curso (Figura 37). Desde aquí se ven las preguntas creadas dentro del curso, además se puede crear más preguntas, ver estadísticas, publicar o despublicar, editar o eliminar el curso.

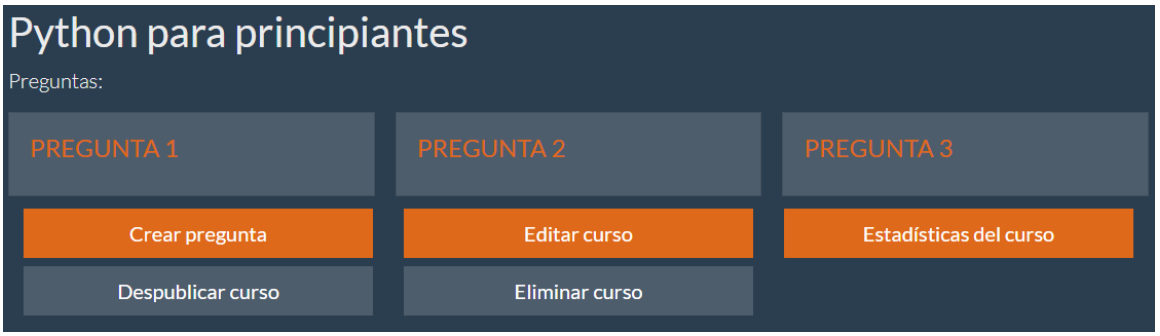


Figura 37. Pantalla de gestión de curso.

7.2.1.8. Publicar o despublicar curso

Al crear un curso este por defecto no será aún accesible. Se requiere que el profesor lo publique. Este paso permite crear las preguntas y definir el curso sin que sea visible para los alumnos. En la pantalla de gestión de curso (Figura 37) se puede publicar un curso que no esté publicado o en su defecto despublicar uno que ya estuviese publicado. La opción correspondiente aparecerá automáticamente dependiendo del estado actual del curso (Figura 38). Como se ve posteriormente en la aplicación móvil solo se muestran los cursos publicados.

```
26      <% if (cursos.publicado===false) { %>
27        <a class="btn btn-secondary btn-lg btn-block" href="/curso/publicarCurso/?id=<%=cursos.id %>" role="button">Publicar curso</a>
28      <% } %>
29      <% if (cursos.publicado===true) { %>
30        <a class="btn btn-secondary btn-lg btn-block" href="/curso/publicarCurso/?id=<%=cursos.id %>" role="button">Despublicar curso</a>
31      <% } %>
```

Figura 38. Código para mostrar el botón de publicar o despublicar curso.

7.2.1.9. Estadísticas de curso

Desde la pantalla de gestión del curso (Figura 37) se puede acceder a las estadísticas de este (Figura 39). En esta pantalla se muestra una lista de todos los alumnos que han completado el curso y la nota que obtuvieron, además de la nota media del curso.

Figura 39. Estadísticas del curso.

La información que se muestra en esta pantalla se obtiene de la colección Completados. El código de la figura 40 muestra la implementación de esta pantalla.

```
<div class="container">
    <div class="container float-left col-md-6">
        <h3>Alumnos que han completado el curso</h3>
        <ul class="list-group">
            <% completados.forEach(function(completados){%>
                <% nota += completados.score %>
                <% i ++ %>
                <li class="list-group-item d-flex justify-content-between align-items-center"><div class="container">
                    <div class="float-left">
                        <a href=""> </a><%= completados.usuario %>
                    </div>
                    <div class="float-right">
                        <span class="badge badge-primary badge-pill"><%= completados.score %> </span>
                    </div>
                </div></li>
            <% })%>
        </ul>
    </div>
    <div class="container float-left col-md-6">
        <h3>Nota media del curso</h3>
        <h3 class="primary"><%= (nota/i).toFixed(2); %> </h3>
    </div>
</div>
```

Figura 40. Código para mostrar estadísticas del curso.

7.2.1.10. Editar curso

Si en la pantalla de gestión de curso se opta por seleccionar la opción editar curso, aparecerá un formulario similar al de la creación de curso (Figura 22). En este formulario de edición (Figura 410) aparecen los datos del curso que estamos editando a modo de marcador de posición (*placeholder*). Al igual que en la edición de usuario (Capítulo 7.2.1.5) no es necesario rellenar el formulario completo, solo los datos a modificar.

Figura 41. Pantalla de editar curso.

7.2.1.11. Eliminar curso

Desde la pantalla de gestión del curso (Figura 37) también es posible eliminar un curso. Al eliminar un curso se eliminan las referencias a este para los usuarios que lo habían desbloqueado. Lo que significa que no podrán volver a realizarlo. La nota obtenida en dicho curso se mantendrá como parte de los puntos totales del alumno. Además en el listado de cursos completados por el alumno seguirá apareciendo dicho curso.

```
//ELIMINAR CURSO
router.get ('/deleteCurso/*',ensureAuthenticated, (req, res)=>
Curso.findOne({_id:req.query.id}, function(err, cursos) {
 res.render('deleteCurso',{
    id:cursos.id
 })
}));
router.get ('/deleteTrue/*',ensureAuthenticated, function(req, res){
    Curso.findById(req.query.id,function(err,cur){
        if(err){
            console.log(err)
        }else{
            if(cur){
                User.find({desbloqueados:cur})
                .then(us=>{
                    us.forEach(usr=>{
                        if(usr){
                            usr.desbloqueados.pull(cur);
                            usr.save();
                        }
                    })
                })
            }
        }

    });
```

Figura 42. Código de eliminar curso.

7.2.1.12. Crear pregunta

Como se ha visto en el Capítulo 7.2.1.7 las preguntas están asociadas a un curso, por lo cual se podrán visualizar dentro de la información relativa a los cursos (Figura 37). Al elegir crear pregunta la aplicación permite generar preguntas de tipo test con 4 opciones para elegir mediante un formulario (Figura 43).

Figura 43. Pantalla de creación de preguntas.

Al crear una pregunta se añade una referencia dentro del array de preguntas del curso desde el cual se está creando la pregunta.

```
//Create Pregunta Handler
router.post('/createPregunta/*',(req,res)=>Curso.findOne({_id:req.query.id}, function(err, cursos){
    const{enunciado,opcion1,opcion2,opcion3,opcion4}=req.body;
    let errors=[];

    if(!enunciado || !opcion1|| !opcion2|| !opcion3|| !opcion4){
        errors.push({msg: 'Rellene todos los campos'})
    }if(errors.length >0){
        res.render('createPregunta',{
            errors,
            enunciado,
            opcion1,
            opcion2,
            opcion3,
            opcion4
        });
    }else{
        const newPregunta=new Pregunta({
            enunciado,
            opcion1,
            opcion2,
            opcion3,
            opcion4
        });
        newPregunta.save();
        cursos.pregunta.push(newPregunta);
        const update=cursos.save();
        console.log(update);
        res.redirect('/curso/?id='+cursos._id);

    }
}));
```

Figura 44. Código de creación de preguntas.

7.2.1.13. Gestión de preguntas

Desde la pantalla de gestión de curso (Figura 37) se puede acceder a las preguntas del mismo. Esta opción muestra la pantalla de gestión de preguntas (Figura 45). Desde esta pantalla es posible editar o eliminar la pregunta.

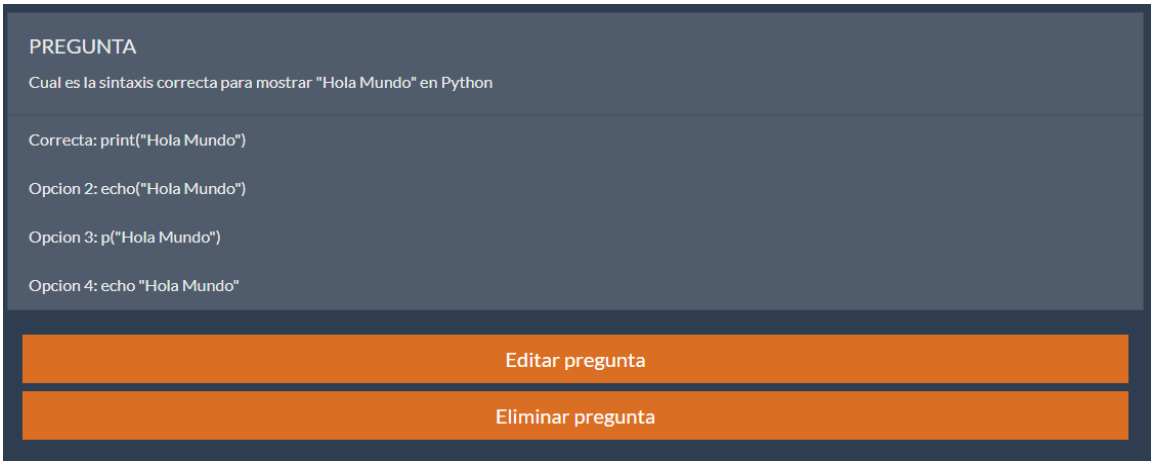


Figura 45. Pantalla de gestión de preguntas.

7.2.1.14. Editar preguntas

Si se accede a la opción de editar pregunta aparecerá un formulario similar al de la creación de pregunta (Figura 43). En este formulario de edición (Figura 46) aparecen los datos de la pregunta que estamos editando a modo de marcador de posición (*placeholder*). Al igual que en la edición de usuario (Capítulo 7.2.1.5) no es necesario rellenar el formulario completo, solo los datos a modificar.

Figura 46. Pantalla de edición de pregunta.

7.2.1.15. Eliminar preguntas

Desde la pantalla de gestión de pregunta(Figura 45) también es posible eliminar una pregunta. Al eliminar una pregunta se eliminan las referencias de esta dentro del curso en el que estaba.

```
17  //ELIMINAR PREGUNTA
18  router.get ('/deletePregunta/*',ensureAuthenticated, (req, res)=>
19  Pregunta.findOne({_id:req.query.id}, function(err, pregunta) {
20   res.render('deletePregunta',{
21      pregunta:pregunta
22   })
23  }));
24  router.get ('/deleteTrue/*',ensureAuthenticated, function(req, res){
25      Pregunta.findOne({_id:req.query.id},function(err,preg){
26          Curso.findOne({pregunta:preg.id},function(err,cur){
27              cur.pregunta.pull(preg)
28              cur.save();
29          });
30          preg.remove();
31          res.redirect('/dashboard');
32      });
33
34  });
```

Figura 47. Código de eliminar pregunta.

7.2.1.16. Registro de un usuario

Para poder usar la aplicación, los usuarios deben registrarse. Para ello desde la pantalla de login pulsará sobre el enlace correspondiente y les conducirá a la pantalla de creación de cuenta (Figura 48). Mediante este proceso se solicitan datos al usuario para luego almacenarlos en la base de datos. Los usuarios registrados desde la aplicación web obtendrán el rol de profesores. Solo los profesores deben llegar a este formulario de registro.

El proceso cuenta con varios niveles de verificación de datos. En primer lugar se comprueba que se han rellenado todos los datos obligatorios. Se verifica que los dos campos de contraseña coinciden para evitar errores. Además se requiere que la contraseña tenga más de 6 caracteres. Posteriormente se comprueba que los datos que identifican al usuario no coinciden con los de otro usuario registrado en el sistema. Si todas estas verificaciones proceden de forma satisfactoria, se continúa a encriptar la contraseña del usuario para no almacenarla como texto plano en base de datos, añadiendo así una capa de seguridad (Figura 49).

Figura 48. Pantalla de creación de cuenta.

```
    //Hash Password
    bcrypt.genSalt(10,(error, salt)=>
    bcrypt.hash(newUser.password,salt, (error, hash)=>{
        if(error) throw err;
        //Convertir contraseña en hash
        newUser.password=hash;
        //Guardar usuario
        newUser.desbloqueados=[];
        newUser.save()
        .then(user=>{
            req.flash('success_msg', 'Ya te has registrado y puedes iniciar session');
            res.redirect('/users/login');
        })
        .catch(err =>console.log(err));

    }))
}
```

Figura 49. Código de encriptación de la contraseña.

7.2.1.17. Inicio de sesión

Para acceder a la aplicación, el usuario utiliza la pantalla de inicio de sesión o *login* que se muestra en la figura 50.

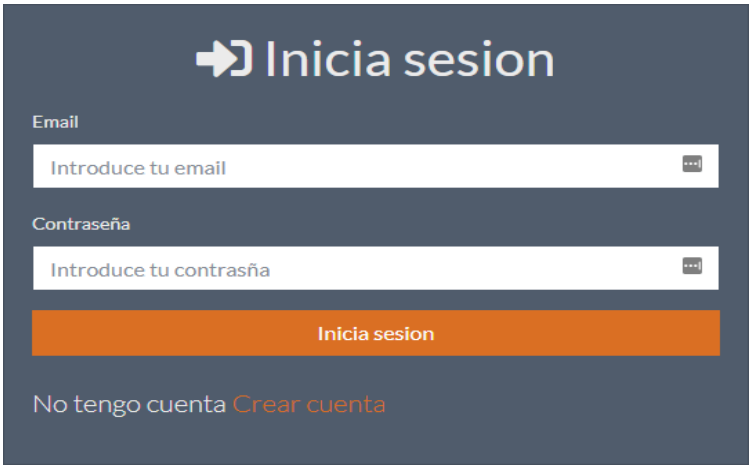


Figura 50. Pantalla de inicio de sesión.

7.3. API REST

En este apartado se especifica el diseño de la API REST que se ha implementado. Las funciones implementadas satisfacen las necesidades de obtener datos de la aplicación móvil. Desde esta se utiliza Retrofit para hacer peticiones HTTP y obtener la información necesaria.

Una de las funciones principales de la API es obtener los cursos que tiene desbloqueado un usuario. Para posteriormente poder mostrarlos en la aplicación móvil (Figura 50).

ENDPOINT	GET	POST	PUT	DELETE
api/registerUser		X		
api/login	X		X	X
api/getCursos	X		X	X
api/getCursosCompletados	X	X		
api/getPreguntas	X		X	X
api/endQuiz		X		

Tabla 1. Endpoints API

Esta función envía una petición POST al servidor con el id del usuario deseado y este devuelve el listado de cursos que tiene desbloqueados. Podemos ver en la figura 51 la implementación en el servidor de la función y en la figura 52 la respuesta del mismo.

```
//GET CURSOS
router.post ('/getCursos/', (req, res)=>{
    var post_data = req.body;
    var id=post_data.id;
    User.findById(id).then(usr=>{
        if(usr){
            Curso.find({_id:usr.desbloqueados}).then(cur=>{
                if(cur){
                    res.json(cur);
                }
            })
        }
    })
}
);
```

Figura 51. Código API para obtener los cursos de un usuario.

Figura 52. Respuesta JSON de la API para obtener los cursos de un usuario.

7.4. Aplicación Android

En este apartado se destacan las funcionalidades específicas de la aplicación móvil. Se incluyen procesos similares a los de la aplicación web como registro, edición o eliminación de cuenta.

7.4.1. Inicio de sesión y registro.

Al ejecutar la aplicación móvil el usuario se encuentra con una pantalla de inicio de sesión (Figura 53). Desde esta pantalla se puede iniciar sesión o registrar un usuario nuevo. Los registros desde la aplicación móvil generan un usuario de tipo alumno.

Figura 53. Pantalla de inicio de sesión móvil.

Si se opta por registrar un usuario nuevo, aparece la pantalla de crear cuenta (Figura 54). Una vez cumplimentados los datos se podrá iniciar sesión.

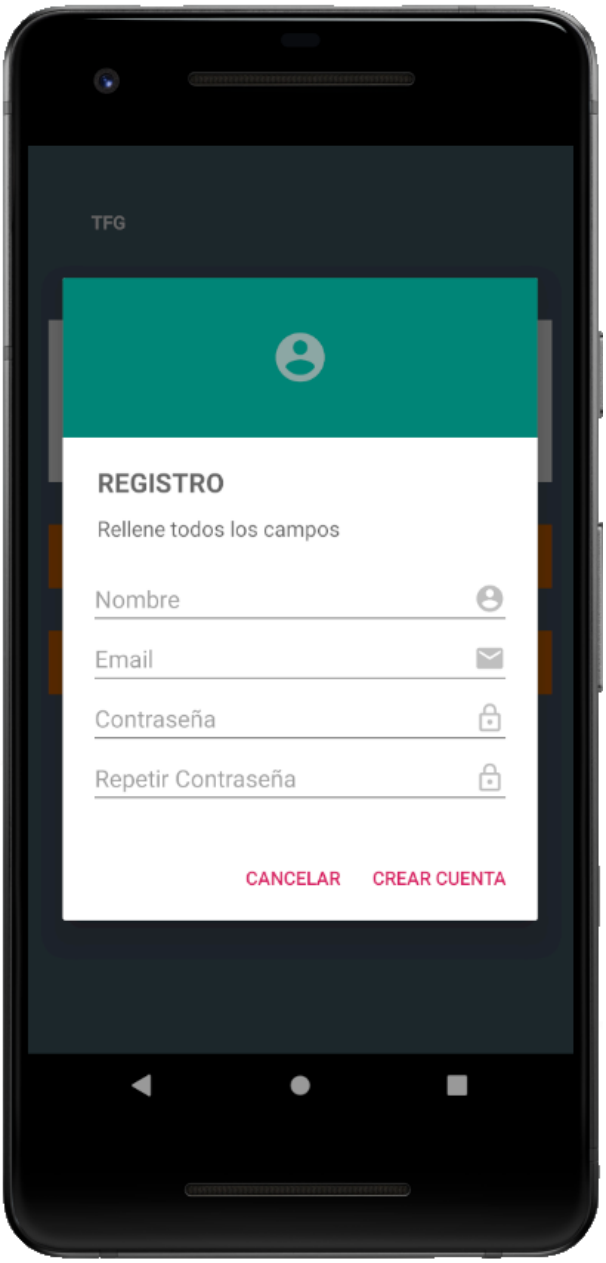


Figura 54. Pantalla de crear cuenta móvil.

7.4.2. Pantalla principal aplicación móvil

Cuando el usuario usuario se autentica en el sistema a través de la aplicación móvil accede a la pantalla principal (Figura 55) en la que se muestran los cursos que ha desbloqueado el usuario. Desde esta pantalla se puede acceder a un curso o cerrar sesión.

Figura 55. Pantalla principal de aplicación móvil.

7.4.3. Iniciar curso

Si se accede a un curso desde la pantalla principal (Figura 55) aparecen directamente hasta diez preguntas aleatorias del curso. Al crear las preguntas (Capítulo 7.2.1.12) la respuesta correcta siempre se introduce en la opción 1, al mostrar la pregunta en la aplicación móvil las respuestas aparecen en orden aleatorio (Figura 56).

Para contestar la pregunta se marca una opción. Una vez marcada, no se puede cambiar la opción elegida. La aplicación muestra un mensaje de retroalimentación diciendo si se ha acertado o no la respuesta. Para continuar se pulsa el botón siguiente.

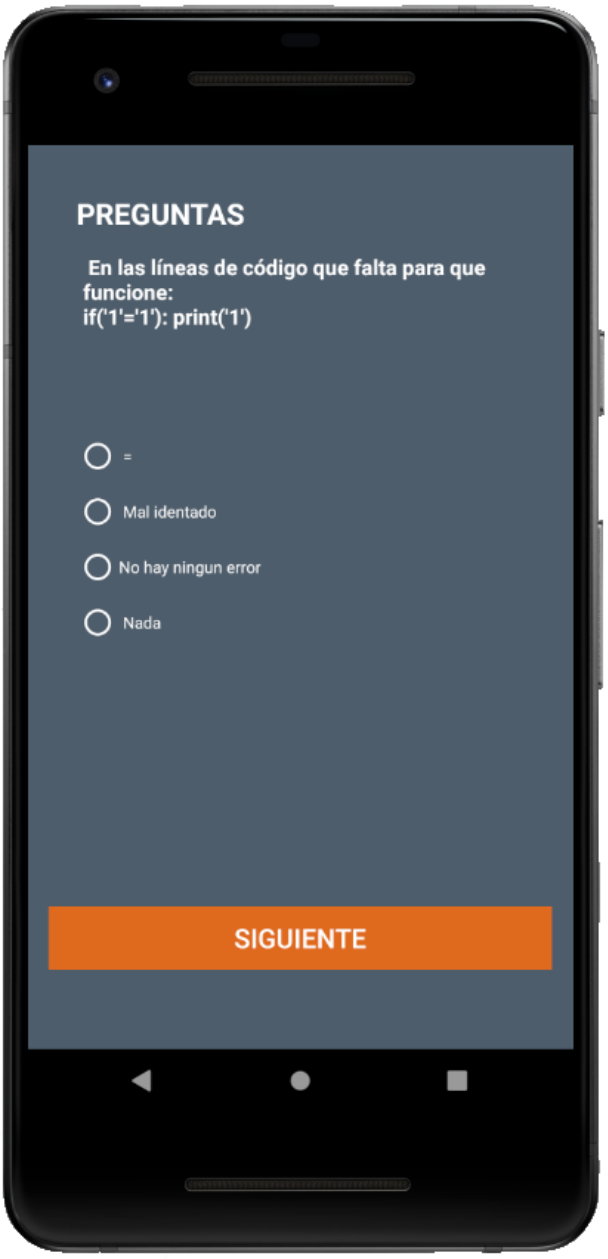


Figura 56. Pantalla de preguntas en la aplicación móvil.

7.4.4. Finalizar curso

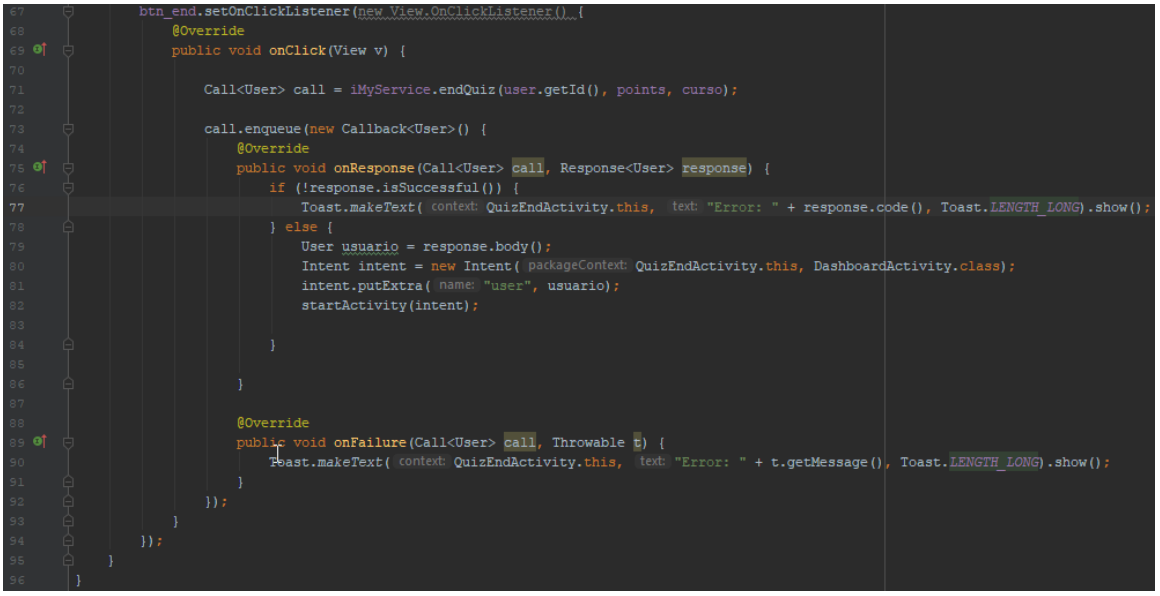
Después de contestar las preguntas de un curso aparecerá una pantalla con la puntuación (Figura 57). La puntuación de un curso se calcula en base a 10, se considera aprobar el curso el obtener una puntuación superior a 5. Al pulsar finalizar se volverá a la pantalla principal (Figura 55) y se habrá desbloqueado el curso que corresponda en función de la puntuación obtenida.

Los alumnos podrán repetir el curso cuantas veces sea necesario pero solo el primer intento suma puntos y desbloquea otro curso. El resto de intentos no cuentan a efectos de puntuación ni desbloquean otro curso si se obtiene un resultado distinto al del primer intento. Los profesores podrán probar los cursos desde la aplicación móvil

pero sus resultados tampoco serán almacenados como puntuación, aunque si desbloquearán el resto de cursos.

Figura 57. Pantalla de finalización de curso.

Al pulsar el botón finalizar se realiza una petición al servidor (Figura 58) enviando el usuario, el curso y la puntuación obtenida. En el servidor se procesa esta petición sumando los puntos al total, actualizando el curso como completado para el usuario y desbloqueando el curso correspondiente (Figura 59). Como respuesta se devuelve al cliente el usuario con los cambios realizados. Así que al finalizar el curso y volver a la pantalla principal el alumno ya podrá ver su nueva puntuación y resultados.



```
btn_end.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        Call<User> call = iMyService.endQuiz(user.getId(), points, curso);

        call.enqueue(new Callback<User>() {
            @Override
            public void onResponse(Call<User> call, Response<User> response) {
                if (!response.isSuccessful()) {
                    Toast.makeText( context: QuizEndActivity.this,  text: "Error: " + response.code(), Toast.LENGTH_LONG).show();
                } else {
                    User usuario = response.body();
                    Intent intent = new Intent( packageContext: QuizEndActivity.this, DashboardActivity.class);
                    intent.putExtra( name: "user", usuario);
                    startActivity(intent);

                }

            }

            @Override
            public void onFailure(Call<User> call, Throwable t) {
                Toast.makeText( context: QuizEndActivity.this,  text: "Error: " + t.getMessage(), Toast.LENGTH_LONG).show();
            }
        });
    }
});
}
}
```

Figura 58. Código de finalización de curso.

```
//END QUIZ
router.post('/endQuiz',(req,res)=>{
    var post_data = req.body;

    var userid=post_data.userid;
    var points=post_data.points;
    var cursoname=post_data.cursoname;

    points=parseInt(points,10);


    User.findById(userid).then(usr=>{
        if(usr){
            Curso.findOne({name:cursoname}).then(cur=>{
                if(cur){
                    var yaCompletado=usr.completados.some(function(ya){
                      return ya.equals(cur._id);
                    });
                    if(!yaCompletado){
                        if(usr.points){
                          usr.points=usr.points+points;
                        }
                        if(points<5){
                            Curso.findById(cur.bUnlock).then(add=>{
                                if(add){
                                    usr.desbloqueados.push(add);
                                }
                            });
                        }else{
                         Curso.findById(cur.aUnlock).then(add=>{
                            if(add){
                               usr.desbloqueados.push(add);
                            }
                         });
                        }
                        usr.completados.push(cur);
                        usr.save().then(function(){
                            console.log("Usuario salvado");
                        }).catch(function(err){
                            console.log(err);
                        });
                        score=points;
                        const comp = new Completados();
```

Figura 59. Código de finalización de curso API.

7.4.5. Opciones de cuenta aplicación móvil

Desde la pantalla principal (Figura 55) se puede acceder a las opciones de la cuenta pulsando en el botón de la esquina superior derecha. En esta pantalla de opciones (Figura 60) se puede editar o eliminar la cuenta del usuario actual. Para editar la cuenta tendremos cuadros de texto donde aparecen los datos del usuario. Se modifican los que se desee cambiar y se pulsa en editar cuenta.

Figura 60. Pantalla de opciones aplicación móvil.

En cambio si lo que se desea es eliminar la cuenta al pulsar en el botón correspondiente aparece una pantalla de confirmación. Si se procede a eliminar, se cierra la sesión enviando al usuario a la pantalla de inicio de sesión y se elimina la cuenta del mismo.

7.4.6. Resultados aplicación móvil

La última opción a la que se puede acceder desde la pantalla principal de la aplicación móvil (Figura 55) es a ver los resultados obtenidos hasta el momento. En la pantalla de resultados el usuario puede ver la puntuación total y cada curso que ha completado con la puntuación obtenida en el mismo (Figura 61).

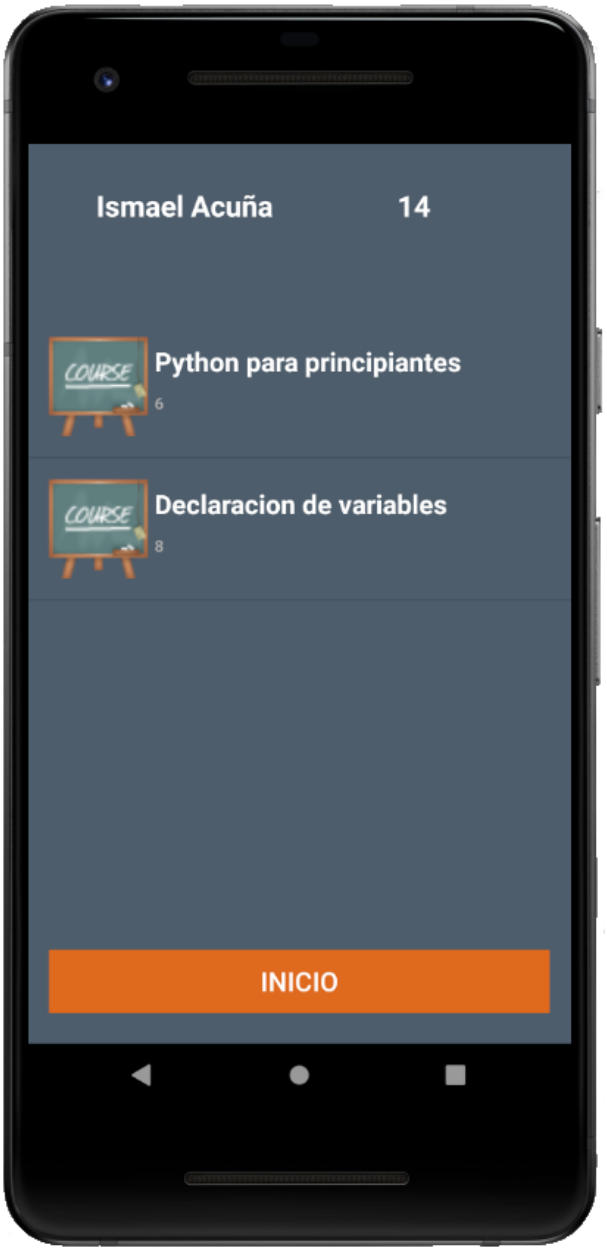


Figura 61. Pantalla de resultados aplicación móvil.

Para obtener esta pantalla, al pulsar en resultados se inicia una actividad nueva en Android que realizará la petición adecuada y mostrará los resultados. En este caso los datos se obtienen de la colección completados. La llamada devuelve una lista de los cursos con sus puntuaciones (Figura 62).

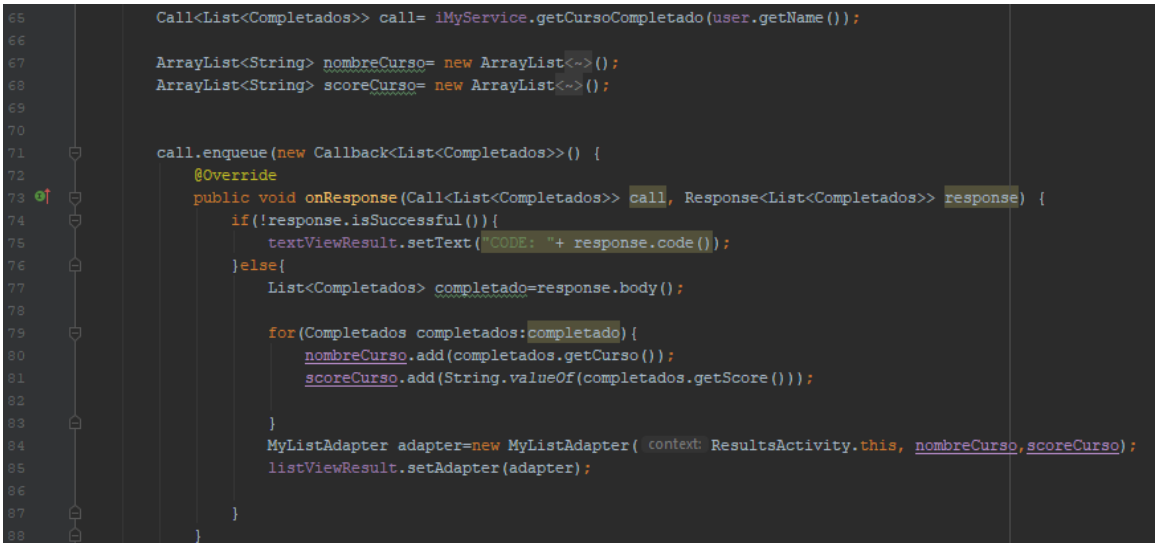


Figura 62. Código petición de resultados aplicación móvil.

7.4.7. Gestión de las peticiones HTTP en la aplicación móvil.

El funcionamiento de la aplicación móvil se basa en obtener los datos de la API y mostrarlos al usuario. A continuación se verá cómo se implementan estas peticiones.

Primeramente se debe configurar Retrofit (Capítulo 3.2.6), que gestionará las peticiones HTTP con el servidor de la API. Además se necesita implementar las clases de modelo para poder serializar las respuestas.

En la configuración de Retrofit se especifica la dirección del servidor donde se encuentra la API. En la figura 63 se puede apreciar dicha especificación, en este caso apuntando a una dirección local y al puerto que corresponda.

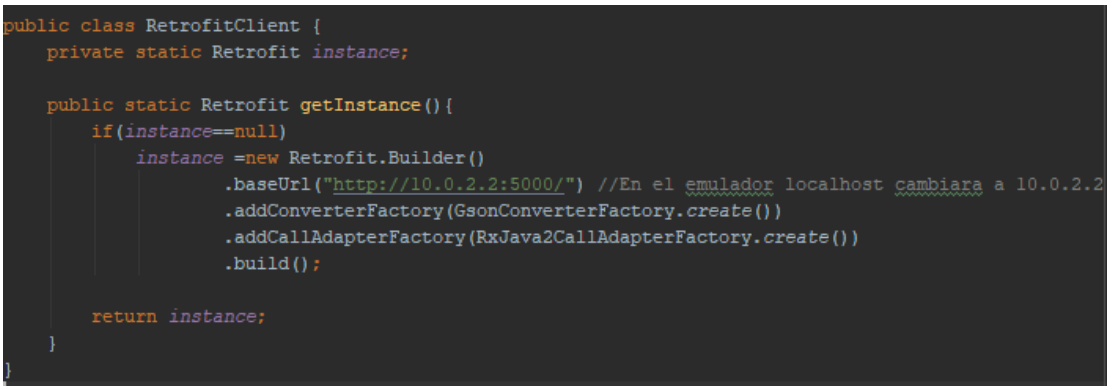


Figura 63. Código de configuración de Retrofit.

También se requiere la definición de los servicios de la API (Figura 64). En estas definiciones se especifican los parámetros de las peticiones y los tipos de datos que se reciben.

```
@POST("api/getCursos")
@FormUrlEncoded
Call<List<Curso>> getCurso(@Field("id")String id);
```

Figura 64. Código de cabecera de las peticiones desde Retrofit.

Se deben implementar las clases de estos tipos de datos con sus correspondientes operaciones (Figura 65).

```
import java.io.Serializable;

public class Curso implements Serializable {
    private String name;
    private String description;
    private String level;
    //Pregunta pregunta[];

    public String getName() { return name; }

    public String getDescription() { return description; }

    public String getLevel() { return level; }
}
```

Figura 65. Código de clases de Modelo en Android.

```
call.enqueue(new Callback<List<Curso>>() {
    @Override
    public void onResponse(Call<List<Curso>> call, Response<List<Curso>> response) {
        if(!response.isSuccessful()){
            textViewResult.setText("CODE: "+ response.code());
        }else{
            List<Curso> cursos=response.body();

            for(Curso curso:cursos){
                titulos.add(curso.getName());
                descripciones.add(curso.getDescription());

            }
            MyListAdapter adapter=new MyListAdapter( context: DashboardActivity.this, titulos,descripciones);
            listViewResult.setAdapter(adapter);


            listViewResult.setOnItemClickListener(new AdapterView.OnItemClickListener() {
                @Override
                public void onItemClick(AdapterView<?> adapterView, View view, int position, long id) {

                    Intent intent = new Intent( packageContext: DashboardActivity.this, QuizActivity.class);
                    intent.putExtra( name: "curso", titulos.get(position));
                    intent.putExtra( name: "user", user);
                    startActivity(intent);
                }
            });
        }
    }
    @Override
    public void onFailure(Call<List<Curso>> call, Throwable t) {
        textViewResult.setText((t.getMessage()));

    }
});
```

Figura 66. Código gestión de las peticiones HTTP en Android.

Al lanzar una petición y una vez obtenida la respuesta el adaptador que se ha definido para Retrofit, en este caso Gson, una biblioteca que permite la serialización y deserialización de objetos Java y su representación JSON. Adapta la respuesta a objetos de las clases modelo definidas. De este modo se utilizan en la aplicación con objetos de usuario, cursos o preguntas.

8. Conclusiones y trabajo futuro

8.1 Conclusiones

Durante este proyecto se ha desarrollado una herramienta que permite a profesores generar cursos y caminos para el aprendizaje de un lenguaje de programación. También permite a los alumnos utilizar estos cursos a través de una aplicación móvil que se puede catalogar en el ámbito de M-learning. Se ha procurado generar interfaces sencillas y fáciles de utilizar para ambos perfiles de usuario.

Se han completado las funcionalidades que permiten al profesor la gestión de los cursos y preguntas de estos, creación, edición y modificación de ambos. De cara a los alumnos de han llevado a término las funcionalidades que permiten completar un curso contestando sus preguntas y así desbloquear cursos siguientes. Estas funcionalidades junto a las de gestión de usuario e inicio de sesión constituyen el núcleo del proyecto por lo que era fundamental su desarrollo. El poder acceder al ranking de alumnos cerraba las funcionalidades que pretendía el proyecto en su inicio.

8.2 Trabajo futuro

Una vez completadas las funcionalidades especificadas en el trabajo, se plantean las siguientes líneas de trabajo futuras:

- **Implementar un sistema de logros.** Actualmente la aplicación tiene como recursos de gamificación el sistema de puntuación y los posibles caminos a desbloquear según las puntuaciones obtenidas en los cursos. Para ampliar estos recursos una mejora a desarrollar sería un sistema de logros que incite al alumno a participar de forma activa.

- **Framework M-learning.** El sistema actualmente permite generar cursos dentro de un mismo marco de conocimiento, Python en este caso. Una línea de trabajo futura sería la posibilidad de añadir una capa superior a estos cursos que englobe varios cursos, el sistema funcionara como un marco de trabajo para adaptar aplicaciones de m-learning a cualquier entorno de enseñanza.

- **Distintos tipos de preguntas.** Actualmente el sistema permite únicamente la creación de preguntas tipo test. Se plantea como mejora añadir distintos tipos de preguntas a nuestros cursos.

- **Carga masiva de datos.** A la conclusión de este proyecto el sistema solo permite la entrada de datos a través de los formularios implementados. Una mejora que se considera como desarrollo futuro es la de poder cargar en el sistema la información relativa a cursos y preguntas de forma masiva desde fuentes externas como pueden ser ficheros CSV u otro formato de datos.

8. Conclusions and future work

8.1. Conclusions

During this project, a tool has been developed that allows teachers to generate courses and itineraries for learning a programming language. It also allows students to use these courses through a mobile app that can be cataloged in the field of M-learning. Efforts have been made to generate simple and easy-to-use interfaces for both user profiles.

During the development of the project concepts and knowledge of software engineering, databases, web application development and mobile application development have been put into practice.

8.2. Future work

Once the functionalities specified in the work have been completed, the following future lines of work are proposed:

- **Implement an achievement system**. Currently the application has as gamification resources the scoring system and the possible paths to unlock according to the scores obtained in the courses. To expand these resources, an improvement to be developed would be an achievement system that encourages the student to participate actively.

- **M-learning Framework**. The system currently allows to generate courses within the same knowledge framework, Python in this case. A future line of work would be the possibility of adding an upper layer to these courses that encompasses several courses, the system will function as a framework to adapt m-learning applications to any teaching environment.

- **Different types of questions**. Currently the system only allows the creation of multiple choice questions. It is proposed as an improvement to add different types of questions to our courses.

- **Mass data loading**. At the conclusion of this project, the system only allows data entry through the implemented forms. An improvement that is considered as a future development is to be able to load information related to courses and questions in bulk from external sources such as CSV files or another data format.

BIBLIOGRAFÍA

[1] «Kahoot!». [En línea]. Disponible en:
https://kahoot.com/what-is-kahoot/
Recuperado el día 17 del 06 de 2020

[2] «Kahoot! Wikipedia». [En línea]. Disponible en:
https://es.wikipedia.org/wiki/Kahoot!
Recuperado el día 17 del 06 de 2020

[3] «Duolingo». [En línea]. Disponible en:
https://es.duolingo.com/
Recuperado el día 17 del 06 de 2020

[4] «Duolingo Wikipedia». [En línea]. Disponible en:
https://es.wikipedia.org/wiki/Duolingo
Recuperado el día 17 del 06 de 2020

[5] «Socrative». [En línea]. Disponible en:
https://socrative.com/
Recuperado el día 17 del 06 de 2020

[6] «Trivinet». [En línea]. Disponible en:
https://www.trivinet.com/es/trivial-online/acerca-de
Recuperado el día 17 del 06 de 2020

[7] «HTML5»[En línea]. Disponible en:
https://developer.mozilla.org/es/docs/HTML/HTML5
Recuperado el día 17 del 06 de 2020

[8] «CSS3»[En línea]. Disponible en:
https://developer.mozilla.org/es/docs/Web/CSS
Recuperado el día 17 del 06 de 2020

[9] «Bootstrap»[En línea]. Disponible en:
https://es.wikipedia.org/wiki/Bootstrap_(framework)
Recuperado el día 17 del 06 de 2020

[10] «Acerca de Node.js®». [En línea]. Disponible en:
https://nodejs.org/es/about/
Recuperado el día 17 del 06 de 2020

[11] «Acerca de NPM». [En línea]. Disponible en:
https://docs.npmjs.com/about-npm/
Recuperado el día 17 del 06 de 2020

[12] «Express.js». [En línea]. Disponible en:
https://en.wikipedia.org/wiki/Express.js
Recuperado el día 17 del 06 de 2020

[13] «EJS». [En línea]. Disponible en:
https://ejs.co/
Recuperado el día 17 del 06 de 2020

[14] «Bcrypt.js». [En línea]. Disponible en:
https://www.npmjs.com/package/bcrypt
Recuperado el día 17 del 06 de 2020

[15] «Retrofit»[En línea]. Disponible en:
https://square.github.io/retrofit/
Recuperado el día 17 del 06 de 2020

[16] «MongoDB» [En línea]. Disponible en:
https://es.wikipedia.org/wiki/MongoDB
Recuperado el día 17 del 06 de 2020

[17] «Android Studio»[En línea]. Disponible en:
https://es.wikipedia.org/wiki/Android_Studio
Recuperado el día 17 del 06 de 2020

[18] «Android Studio »[En línea]. Disponible en:
https://developer.android.com/studio/intro
Recuperado el día 17 del 06 de 2020

[19] «GIT»[En línea]. Disponible en:
https://git-scm.com/about
Recuperado el día 17 del 06 de 2020

[20] «Visual Studio Code»[En línea]. Disponible en:
https://code.visualstudio.com/docs
Recuperado el día 17 del 06 de 2020

[21] «Mongoose»[En línea]. Disponible en:
https://mongoosejs.com/docs
Recuperado el día 17 del 06 de 2020

[22] «Cliente-servidor»[En línea]. Disponible en:
https://es.wikipedia.org/wiki/Cliente-servidor
Recuperado el día 17 del 06 de 2020

[23] «Tema para Bootstrap»[En línea]. Disponible en:
https://bootswatch.com/superhero/
Recuperado el día 17 del 06 de 2020

ANEXOS

El proyecto se encuentra almacenado en dos repositorios. Uno contiene el proyecto de Android Studio para la aplicación móvil y el otro el proyecto de NodeJs para la aplicación web. Ambos repositorios tienen visibilidad pública.

https://github.com/fbofill/tfg-Mobile-ClientSide
https://github.com/fbofill/tfg-Web-ServerSide

ANEXO I: GUÍA DE USO.

En este anexo se incluye una guía de uso que explica cada una de las funciones que puede realizar la aplicación. Primeramente se explican las funcionalidades de la aplicación web y posteriormente las de la aplicación móvil.

1. Aplicación WEB.

1.1. Registro de profesor e inicio de sesión.

Para poder acceder a las funcionalidades de la aplicación el primer paso es crear una cuenta.

Al acceder a la dirección de la aplicación web se encuentra una pantalla donde se elige entre iniciar sesión y crear una cuenta (Figura 67).

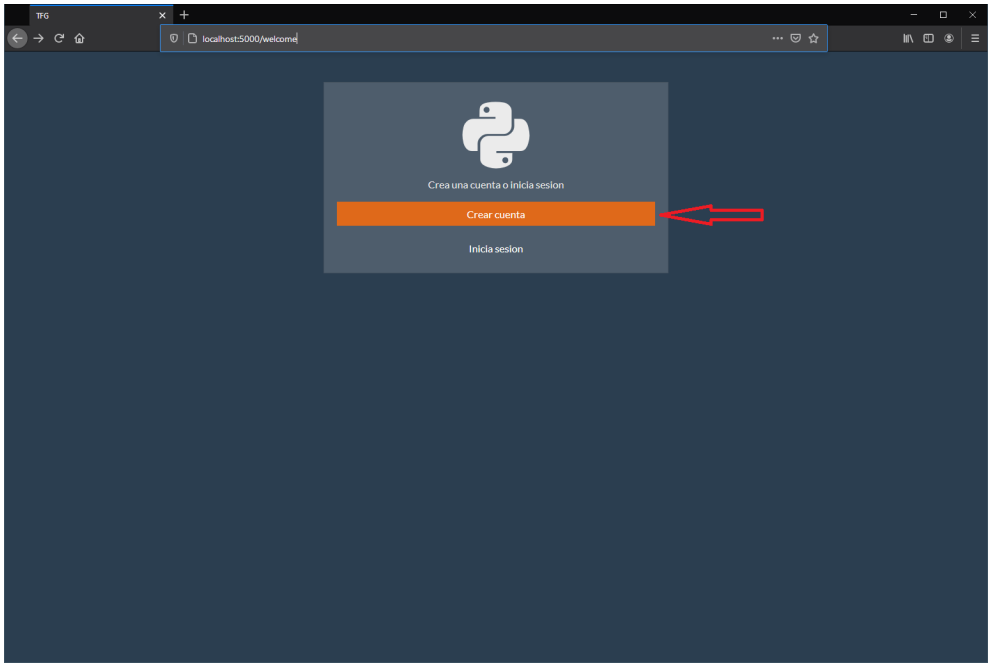


Figura 67. Primera pantalla.

Si se pulsa en la opción de crear una cuenta la aplicación entra en la ruta *users/register/* y muestra el formulario de registro (Figura 68).

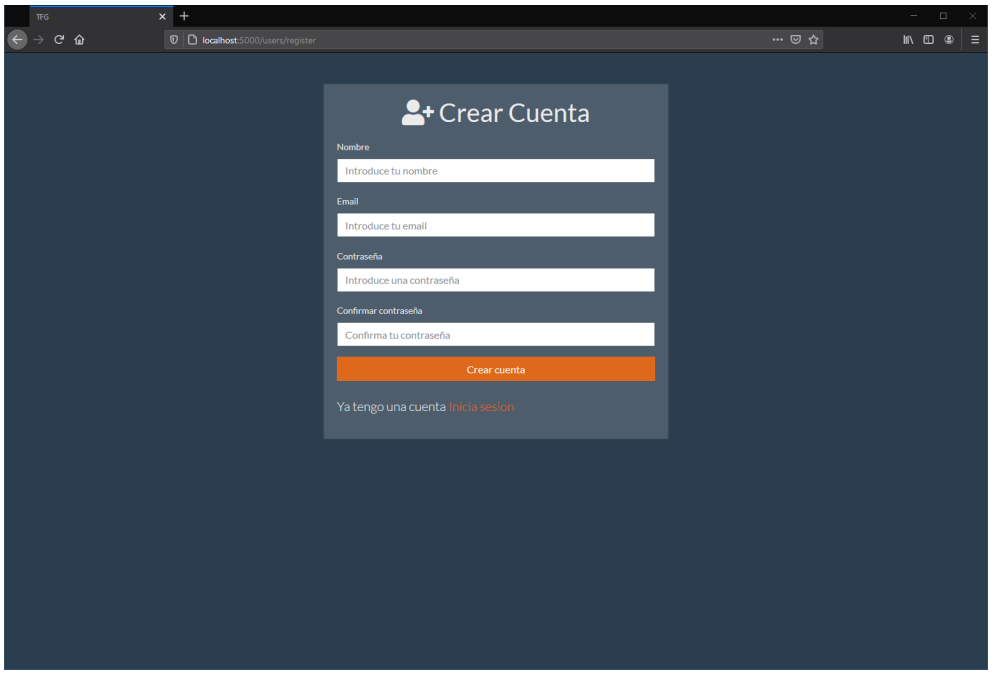


Figura 68. Pantalla de registro.

En esta pantalla se completan los campos requeridos y nos llevará a la pantalla de inicio de sesión (Figura 69).

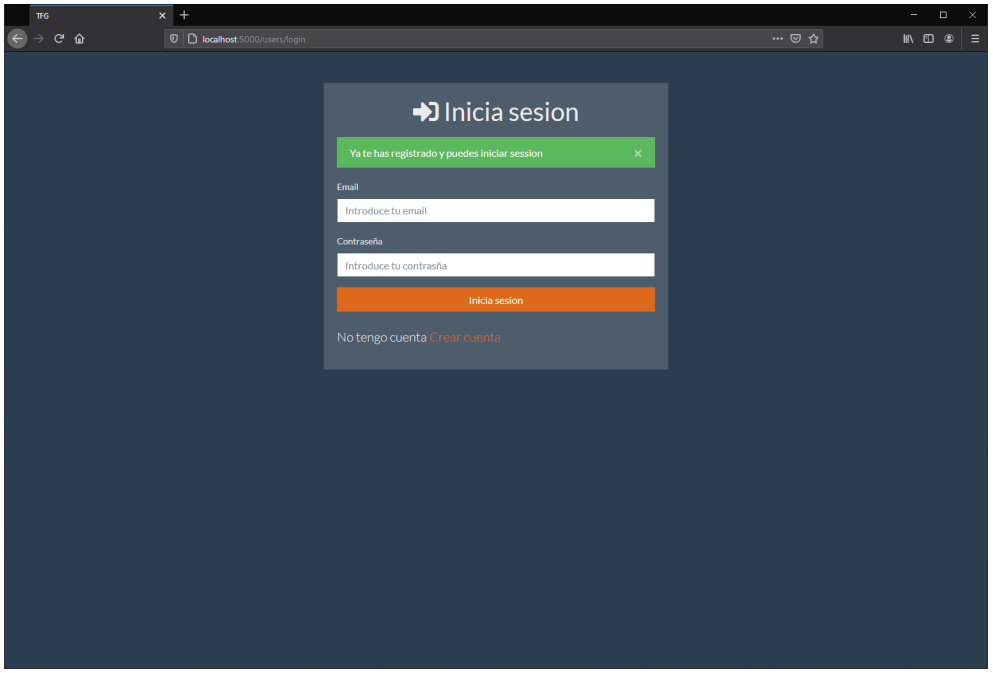


Figura 69. Pantalla de inicio de sesión.

Si se introducen unas credenciales correctas se inicia sesión un usuario profesor que será redirigido a la pantalla principal de la aplicación (Figura 70).

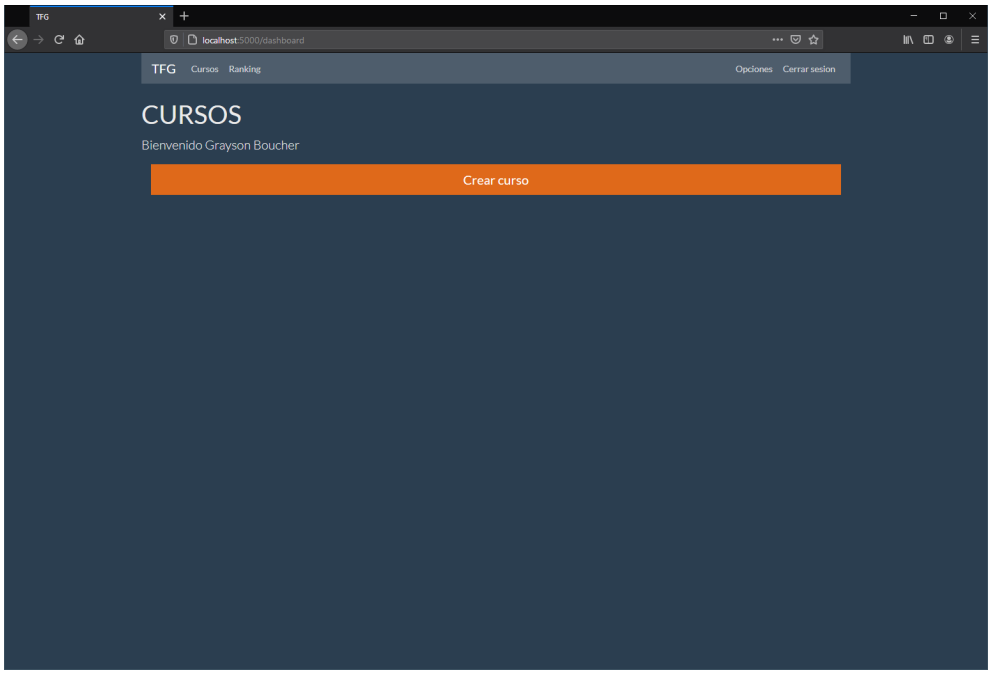


Figura 70. Pantalla principal.

1.2. Gestión de cursos.

En esta pantalla se muestran los cursos creados (ninguno en este ejemplo). Se procederá a crear tres cursos. Para ello al pulsar sobre *Crear curso* se redirige al formulario de creación de cursos.

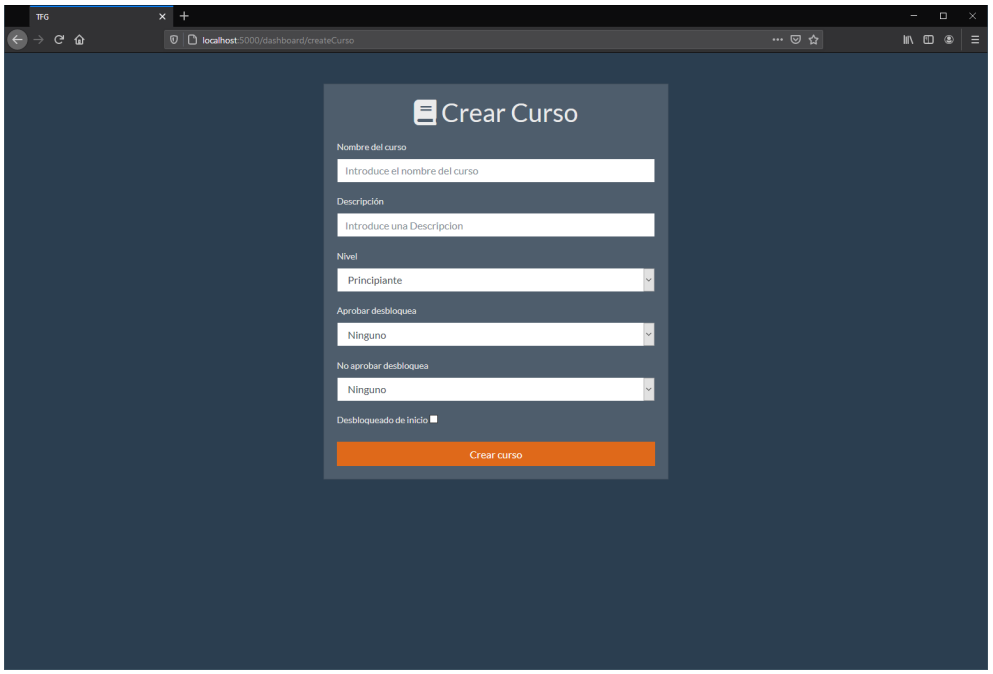


Figura 71. Pantalla crear curso.

En este formulario se deben rellenar los campos de *nombre*, *descripción* y *nivel* del curso. Los otros tres campos *Aprobar desbloquea*, *No aprobar desbloquea* y *Desbloqueado de inicio* son útiles para establecer los caminos a seguir.

Se creará un primer curso con nivel principiante y marcada la casilla Desbloqueado de inicio. Un segundo curso con nivel intermedio y desmarcada la casilla mencionada anteriormente y un tercer curso de nivel principiante y también desmarcado el desbloqueo de inicio. Para todos se dejan *Aprobar desbloquea* y *No aprobar desbloquea* como *Ninguno*.

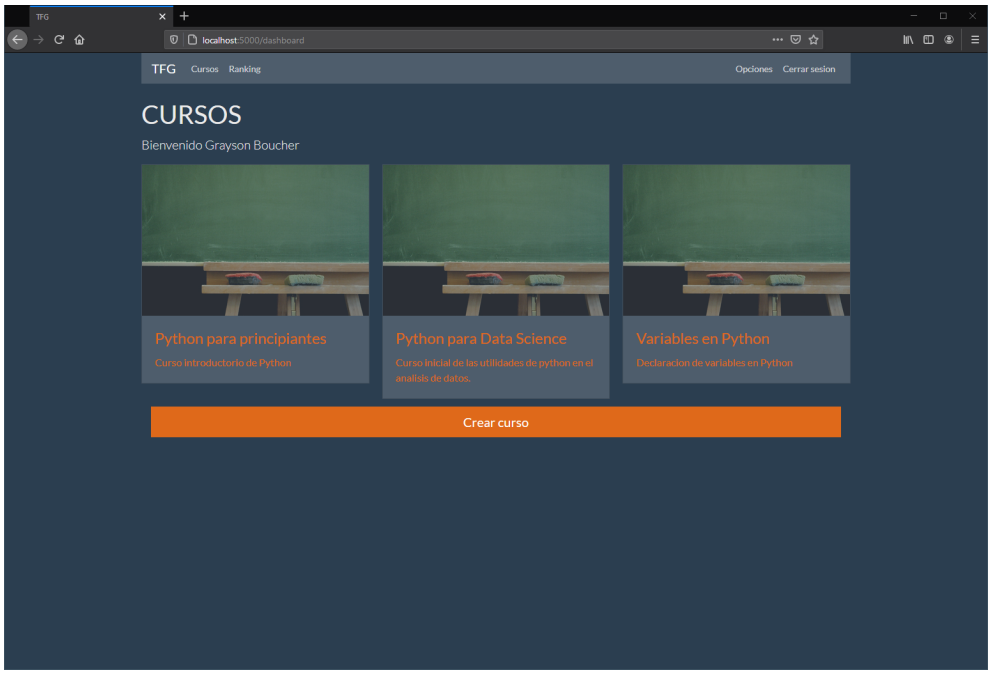


Figura 72. Pantalla inicial con cursos.

El resultado de estas operaciones se puede observar en la figura 72. En este punto ninguno de estos cursos está publicado ni es accesible desde la aplicación móvil.

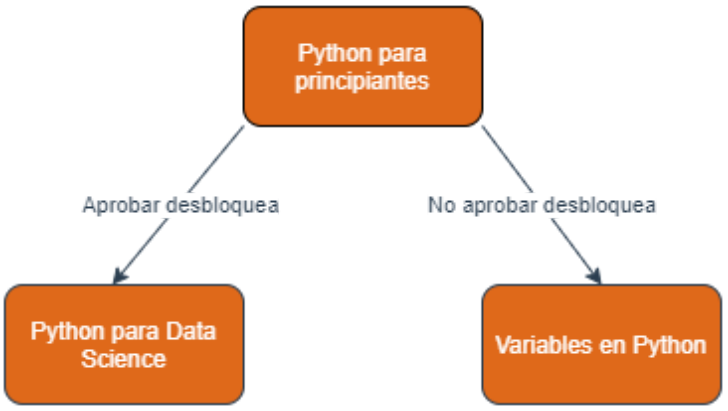


Figura 73. Estructura de cursos.

Antes de publicar, se crea una pequeña estructura de árbol. Editando el primer curso y eligiendo en Aprobar desbloquea el curso de nivel intermedio y en No aprobar desbloquea el otro curso de nivel principiante. Se obtiene una estructura como la de la figura 73.

Para editar los cursos desde la pantalla principal se entra en un curso y se selecciona *Editar curso* (Figura 74). Esto redirige la aplicación al formulario de edición de curso.

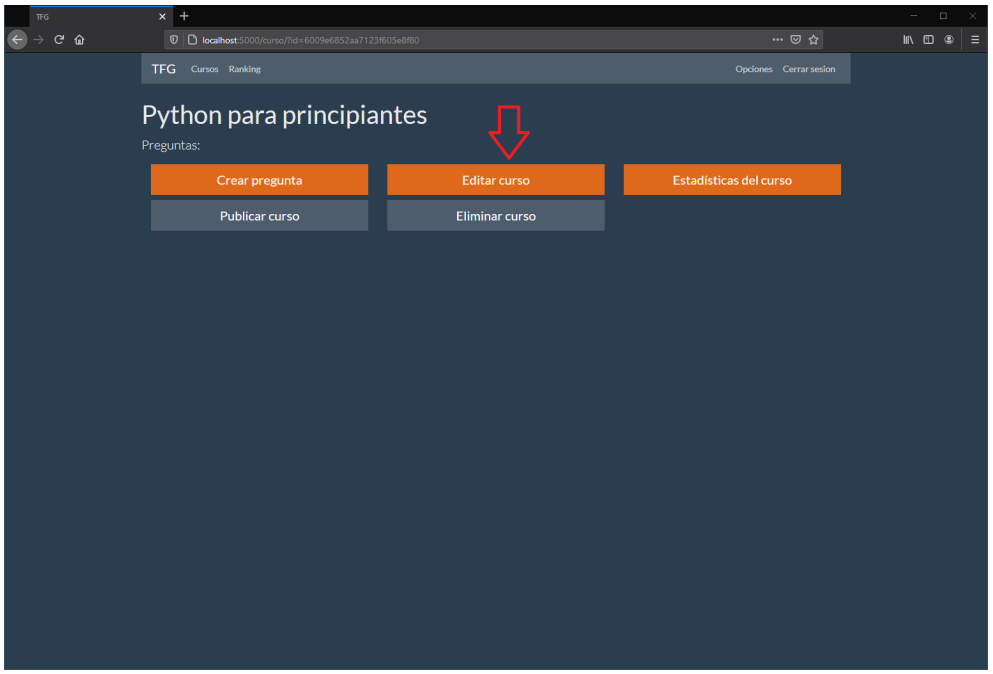


Figura 74. Pantalla de gestión de curso.

1.3. Gestión de preguntas.

El siguiente paso en el funcionamiento normal de la aplicación será poblar los cursos con preguntas. Para ello dentro de la pantalla de gestión de curso se selecciona *Crear Pregunta*. Esto redirige al formulario de creación de preguntas (Figura 75).

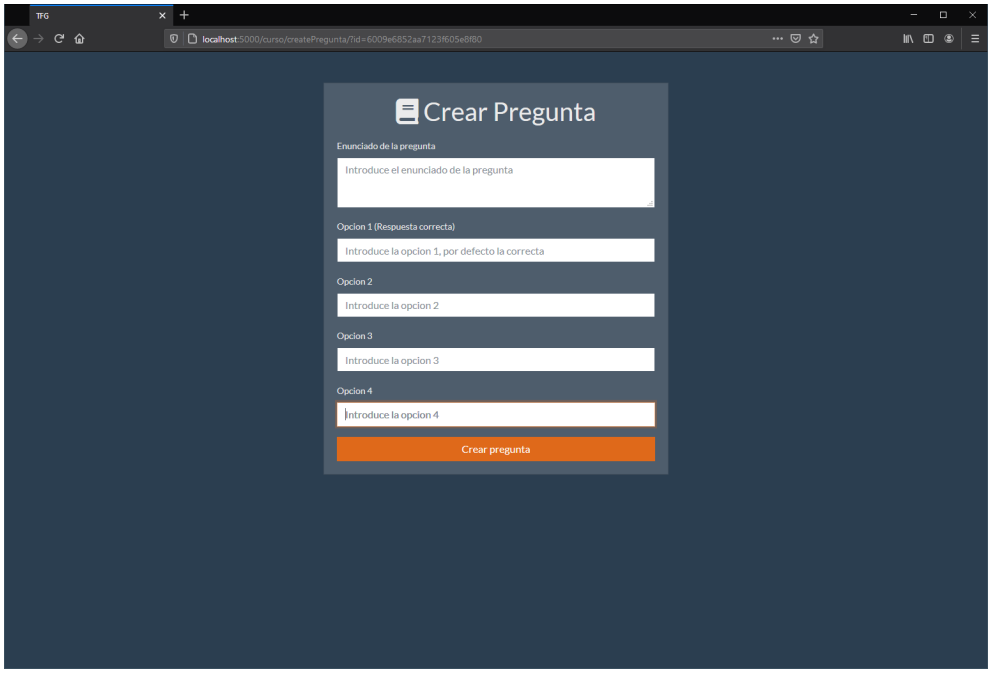


Figura 75. Pantalla de creación de preguntas.

Las preguntas creadas se pueden visualizar dentro del curso (figura 76).

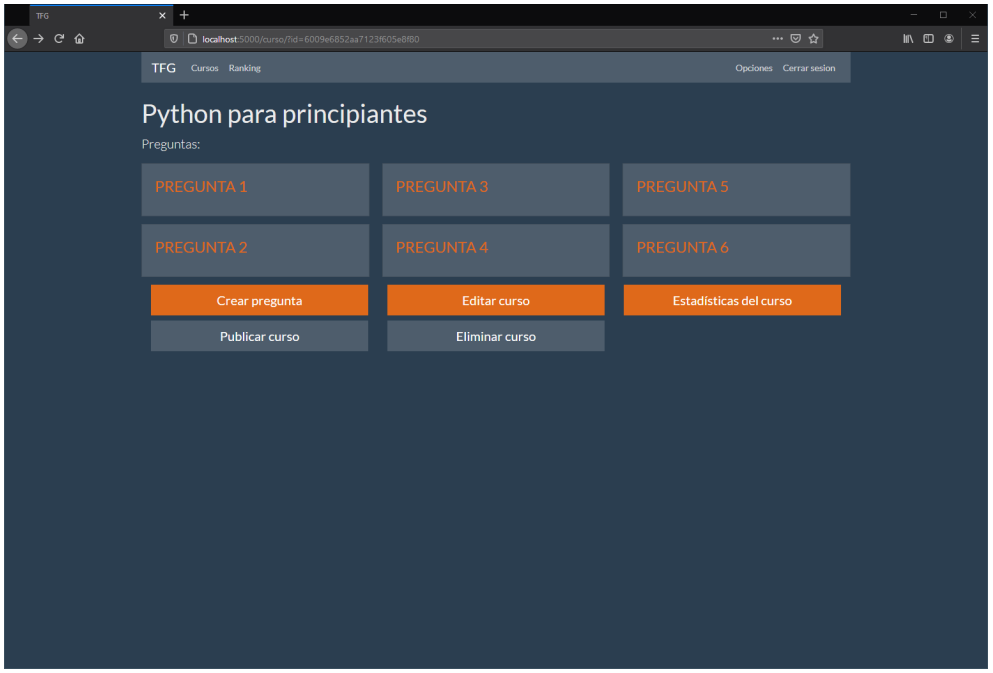


Figura 76. Pantalla de curso con preguntas.

Al acceder a una de estas preguntas se encuentra la pantalla de gestión de preguntas. Desde la cual se puede modificar o eliminar la pregunta (Figura 77).

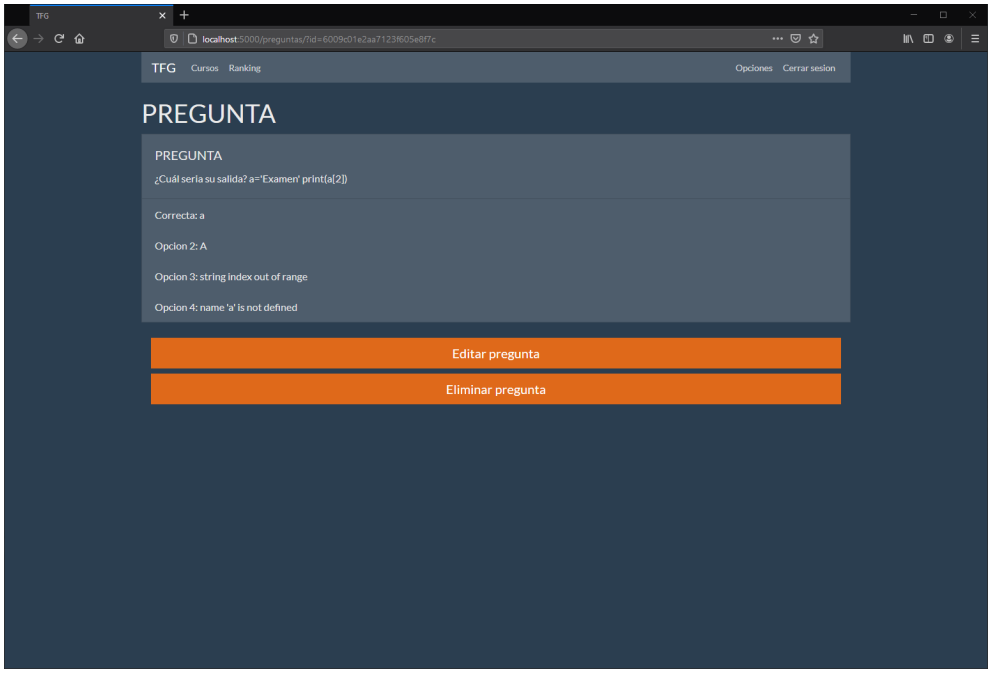


Figura 77. Pantalla de gestión de preguntas.

1.4. Publicar cursos.

Una vez creadas las preguntas dentro de los cursos se pueden publicar los mismos. Para ello desde la pantalla de gestión de cursos (Figura 76) habrá que seleccionar la opción *Publicar curso*.

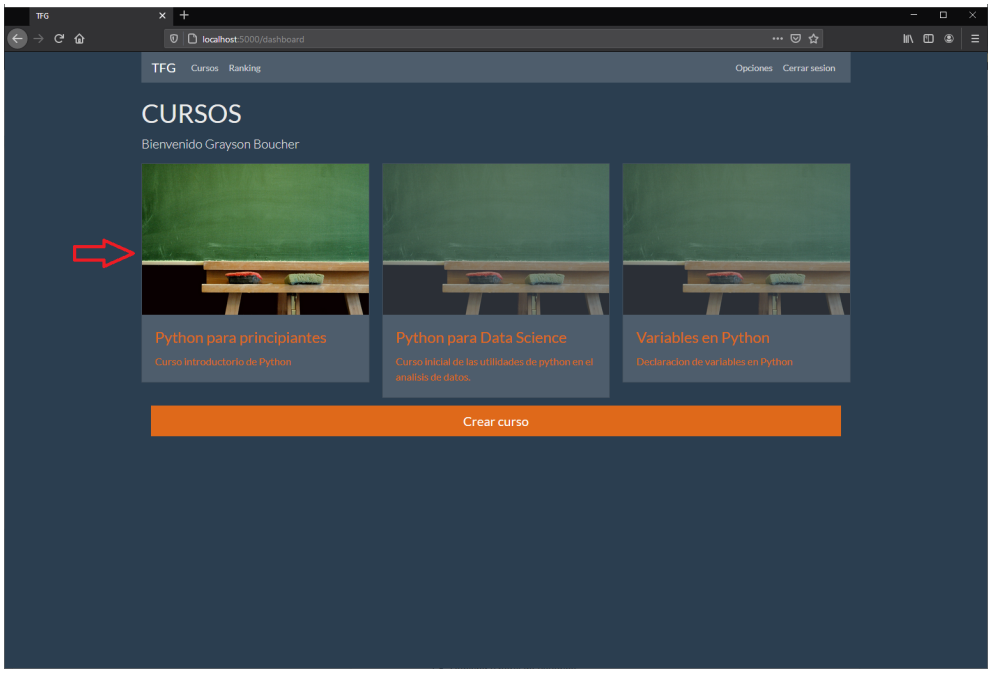


Figura 78. Pantalla de curso publicado.

Los cursos publicados se distinguen en la pantalla principal porque se muestran con más opacidad que los no publicados.

1.5. Ranking y perfil de usuarios.

Desde la pantalla principal se puede acceder a la opción ranking (Figura 79).

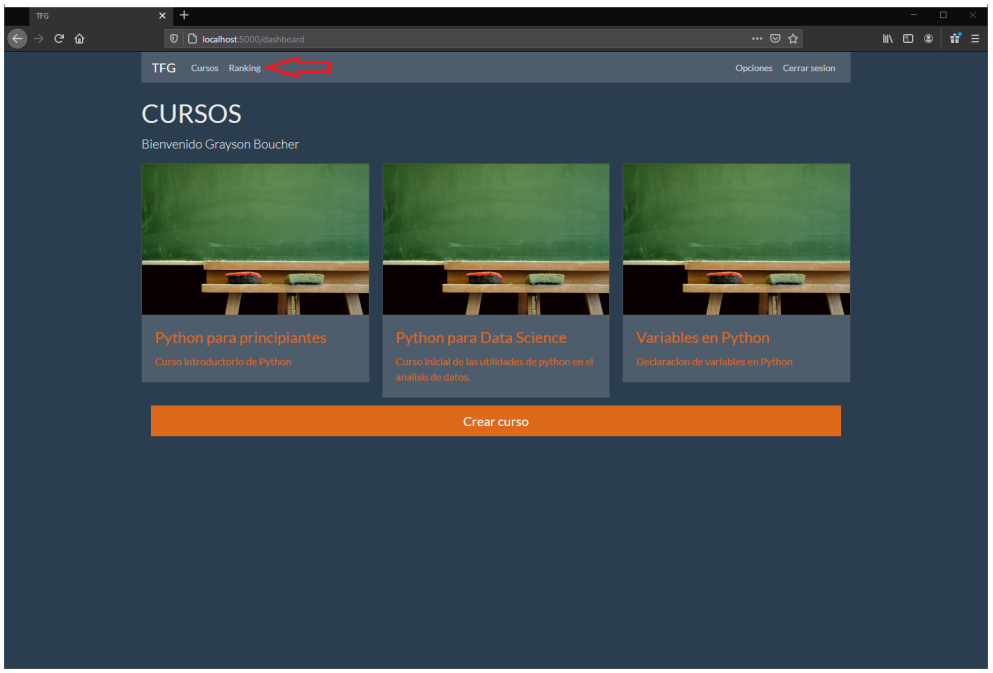


Figura 79. Opción ranking.

Esta opción muestra el listado de alumnos con sus puntuaciones en orden descendente (Figura 80).

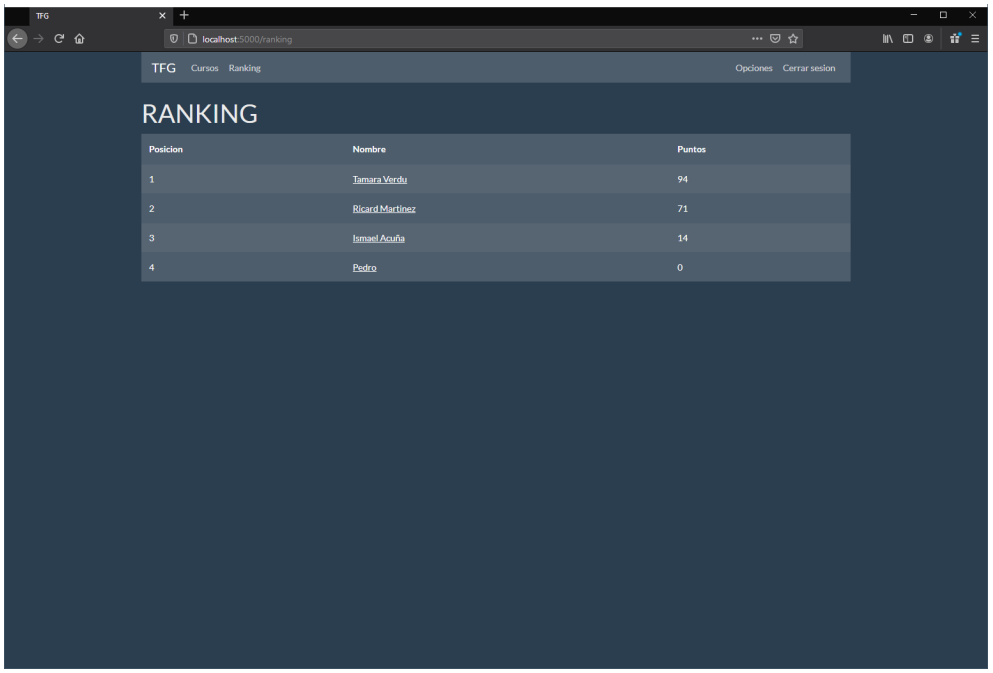


Figura 80. Pantalla ranking.

Al pulsar en el nombre de algún alumno se puede acceder al perfil de usuario de este (Figura 81). El perfil de usuario muestra los datos del usuario como Nombre o email y también la puntuación total y una lista con los cursos que ha realizado y la puntuación obtenida en estos.

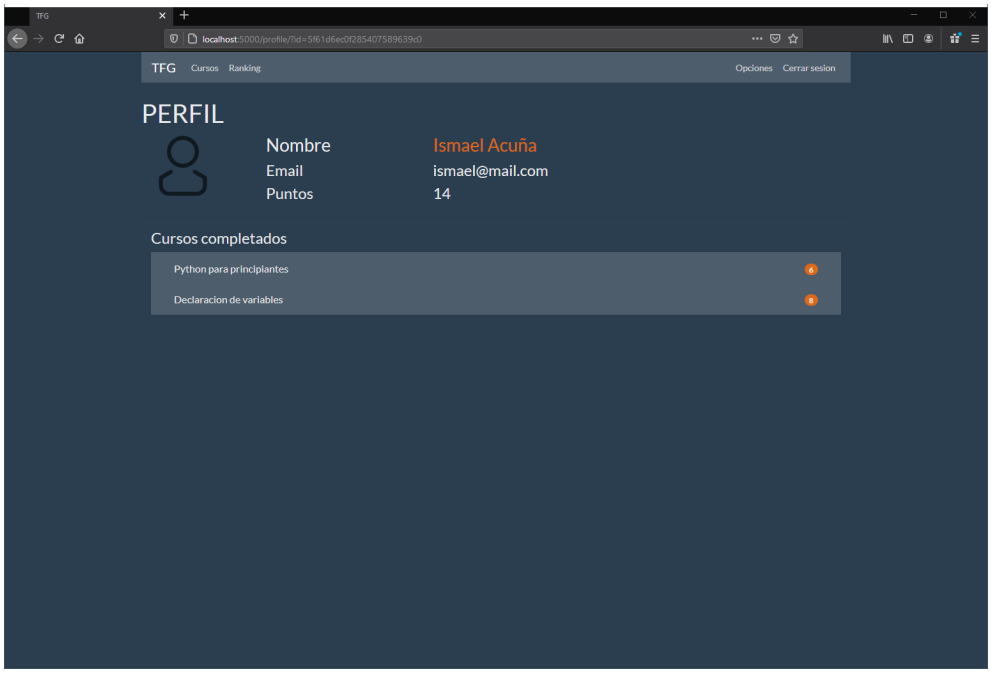


Figura 81. Pantalla perfil de usuario.

Cuando se desee salir de la aplicación, desde cualquier pantalla se debe pulsar la opción cerrar sesión.

2. Aplicación Móvil.

2.1. Registro de profesor e inicio de sesión.

Al iniciar la aplicación móvil se muestra la pantalla de inicio de sesión (Figura 82). Desde aquí para crear una cuenta nueva se ha de acceder a la opción crear cuenta.

Figura 82. Pantalla inicio móvil.

Desde aquí se redirige la aplicación al formulario de crear cuenta desde el cual introduciendo todos los datos se crea una cuenta de alumno nueva.

2.2. Completar un curso.

Al iniciar sesión se ve la pantalla principal de la aplicación móvil (Figura 83). En esta se muestran los cursos que tiene el usuario desbloqueados. De la pequeña estructura de árbol que se creo que el capítulo anterior se puede ver únicamente el curso que estaba marcado como desbloqueado de inicio.

Figura 83. Pantalla principal móvil.

Para completar un curso se pulsa sobre él y automáticamente se lanza el cuestionario correspondiente (Figura 84).

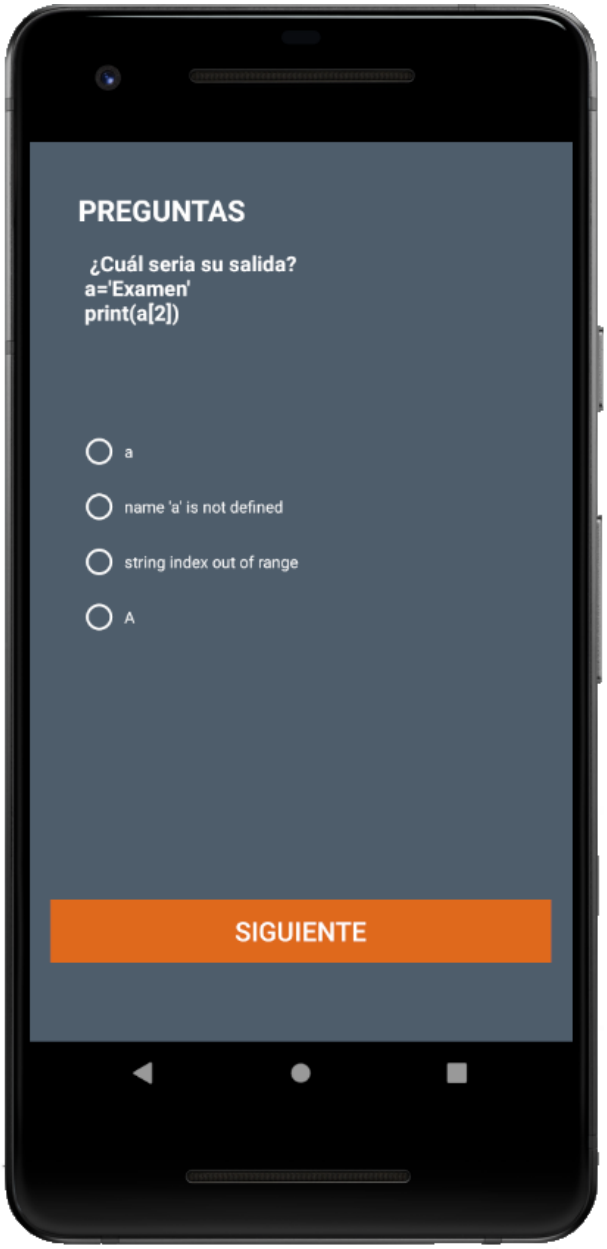


Figura 84. Pantalla de pregunta móvil.

Al terminar el cuestionario aparece la pantalla de puntuación. Pulsando finalizar en esta se vuelve a la pantalla principal pero esta vez estará desbloqueado el curso que corresponda según la puntuación obtenida (Figura 85).

Figura 85. Pantalla de inicio curso desbloqueado móvil.

2.3. Ver resultados.

Desde la pantalla principal (Figura 83) se puede acceder también a la lista de resultados. Aquí se muestra un listado de los cursos completados y la puntuación obtenida en estos (Figuera 86).

Figura 86. Pantalla de resultados móvil.

2.3. Editar o eliminar cuenta.

Las opciones de modificación de cuenta son accesibles desde el botón de la parte superior derecha de la pantalla principal (Figura 85). Desde esta pantalla se puede modificar o eliminar la cuenta eligiendo la opción que corresponda (Figura 87).

Figura 87. Pantalla de opciones de cuenta móvil.

ANEXO II: GUÍA DE INSTALACIÓN.

La ejecución del proyecto requiere la instalación de algunas dependencias. Los dependencias que han de ser instaladas son:

- NodeJs
- Mongodb
- Android Studio

Recomendable:

- Mongo Compass

Una vez instaladas las dependencias accederemos a los repositorios (Figura 49) en el botón Code es posible descargar el proyecto en un archivo ZIP o clonarlo desde una terminal de git con el comando **git clone** "*direccion_del_proyecto*"

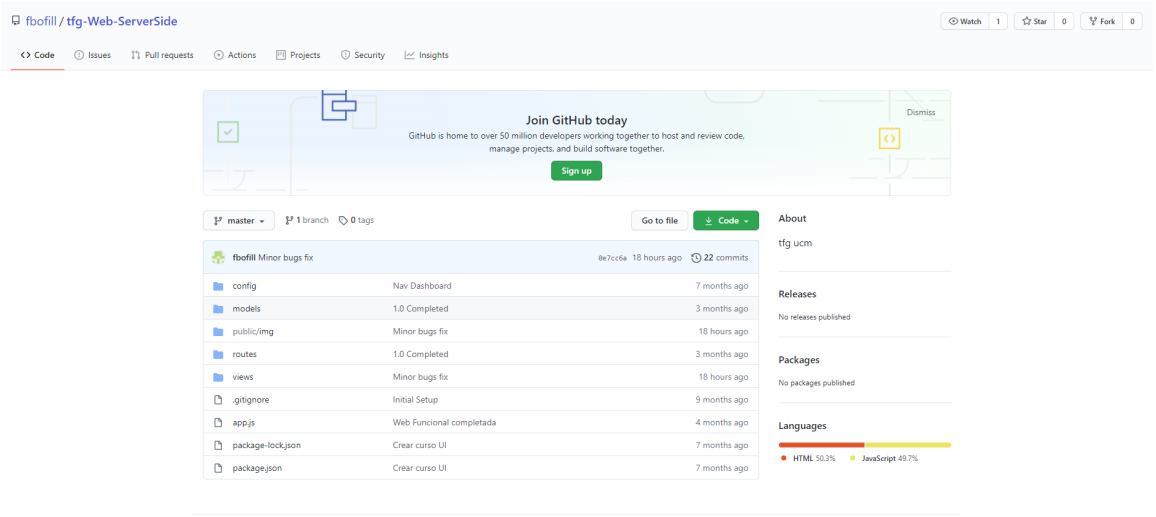


Figura 88: Repositorio del proyecto

Después de descargar los proyectos y descomprimir los ficheros ZIP en el caso que fuese necesario, se podrá proceder a la ejecución de estos.

Aplicación Web y Servidor

Comenzaremos por lanzar el servidor web de Node. Desde una terminal de comandos y una vez ubicados en la carpeta del proyecto web, ejecutaremos la siguiente directiva:

npm install

Esta instalará todas las dependencias necesarias (no incluidas en el repositorio). En caso de que no esté lanzado el servidor de MongoDB lo lanzaremos. Si al instalar MongoDB hemos elegido que se instale como un servicio del sistema operativo se lanzará al iniciar el mismo, en caso contrario, para lanzar en Windows desde la dirección de instalación predeterminada se ejecuta:

"C:\Program Files\MongoDB\Server\4.2\bin\mongod.exe" --dbpath="c:\mongodata\db"

Teniendo lanzado el servidor de MongoDB e instalado las dependencias de paquetes de npm, se procede a ejecutar el siguiente comando para lanzar el servidor Node.js

node app.js

Con esto lanzaremos el servidor, de forma predeterminada en localhost:5000. Si por alguna razón necesitamos cambiar el puerto

```
const PORT = process.env.PORT || 5000;
```

Figura 89: Puerto del servidor Node

Lo podremos hacer editando esta línea en app.js. Una vez lanzados ambos servidores podremos acceder a la aplicación web desde.

http://localhost:5000/

Aplicación Móvil (Cliente)

Posteriormente se procede a lanzar la aplicación móvil que utilizará este servidor web para gestionar las peticiones de la API REST por lo cual el servidor debe estar lanzado para utilizar correctamente la aplicación móvil. Desde android studio configuraremos un dispositivo virtual para lanzar la app (Este proyecto se ha probado íntegramente en un dispositivo Pixel 2 con la configuración por defecto). Si se ha editado el puerto del servidor, tendra que ser modificado también en RetrofitClient.java

```
public static Retrofit getInstance(){
    if(instance==null)
        instance =new Retrofit.Builder()
                .baseUrl("http://10.0.2.2:5000/") //En el emulador localhost cambiara a 10.0.2.2
                .addConverterFactory(ScalarsConverterFactory.create())
                .addCallAdapterFactory(RxJava2CallAdapterFactory.create())
                .build();

    return instance;
}
```

Figura 90: Configuración de Retrofit

.